AD-A091 177 STANFORD UNIV CA DEPT OF COMPUTER SCIENCE F/G 9/2
PROTOTYPES AND PRODUCTION RULES: A KNOWLEDGE REPRESENTATION FOR--ETC(U)
AUG 80 J S AIKINS MDA903-77-C-0322
UNCLASSIFIED STAN-CS-80-814 NL

1 OF 2
AD
A091177

Stanford Heuristic Programming Project
Memo HPP-80-17

August 1980

Department of Computer Science
Report No. STAN-CS-80-814

LEVEL II

PROTOTYPES AND PRODUCTION RULES: A KNOWLEDGE REPRESENTATION FOR COMPUTER CONSULTATIONS

by

Janice S. Aikins

Research sponsored by

Advanced Research Projects Agency

COMPUTER SCIENCE DEPARTMENT
Stanford University

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

9 Doctoral thesis

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER STAN-CS-80-814 (HPP-80-17)	2. GOVT ACCESSION NO. AD-A091 177	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 Prototypes and Production Rules: A Knowledge Representation for Computer Consultations.		5. TYPE OF REPORT & PERIOD COVERED technical, August 1980
		6. PERFORMING ORG. REPORT NUMBER 14 STAN-CS-80-814, (HPP-80-17)
7. AUTHOR(s) 10 Janice S./Aikins		8. CONTRACT OR GRANT NUMBER(s) 15 MDA 903-77-C-0322, PHS-RR-1 -1
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Stanford University Stanford, California 94305 USA		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 12 110
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency Information Processing Techniques Office 1400 Wilson Avenue, Arlington, Virginia 22209		12. REPORT DATE 11 August 1980 / 13. NO. OF PAGES 204
		15. SECURITY CLASS. (of this report) Unclassified
14. MONITORING AGENCY NAME & ADDRESS (if diff. from Controlling Office) Mr. Philip Surra, Resident Representative Office of Naval Research, Durand 165 Stanford University		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(see other side)

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19. KEY WORDS (Continued)

20 ABSTRACT (Continued)

This thesis presents a system called CENTAUR, which demonstrates the effectiveness of representing prototypical knowledge in a combination of frames and production rules for performing computer consultations. Key knowledge representation and control structure problems in *production rule* systems similar to MYCIN are identified, and a set of important characteristics of the structures used for representing problem-solving knowledge is given. CENTAUR's frames, or prototypes, complement the production rules to satisfy these characteristics and represent expected patterns of data that permit a more focused, hypothesis-directed approach to problem solving.

Among the characteristics identified as desirable in the representation structures are the ability to *explicitly* represent (a) prototypical cases, (b) the *context* in which knowledge is applied, and (c) the *strategies* for applying that knowledge. CENTAUR's prototypes consist of patterns of knowledge in the domain which serve as broad contexts, guiding the more detailed processing of the production rules. Strategies for the consultation, or control knowledge, are represented in the prototypes separately from other kinds of domain knowledge. This allows the domain expert to specify control knowledge that is specific to each prototype. Examples are presented which demonstrate how this explicit representation facilitates explanations of the system's reasoning. Further, the organization of knowledge in CENTAUR provides a useful framework for acquiring new knowledge.

CENTAUR has been applied to the domain of pulmonary (lung) physiology in which it provides interpretations of pulmonary function tests. The prototypes represent standard pulmonary disease patterns, and the production rules serve as a stylized form of *attached procedure*. At the highest level, the stages of the consultation itself are represented in a *Consultation* prototype. Thus the advantages of explicit representation apply to control of the consultation process as well.

Other important features of the representation include the use of prototypes as a standard of comparison in order to detect inconsistent or erroneous data, and the representation in production rules of domain expertise to deal with data discrepancies and diagnosis refinement.

Several experiments demonstrating the flexibility of the representation have also been performed. These include the implementation of different top-level prototype selection strategies (*confirmation*, *elimination*, and *fixed-order*), and the use of a second high-level prototype which can review knowledge stored in the domain-level prototypes.

DD FORM 1 JAN 73 1473 (BACK)

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Computer Science Department
Report No. STAN-CS-80-814

Heuristic Programming Project
Memo HPP-80-17

August 1980

PROTOTYPES AND PRODUCTION RULES: A KNOWLEDGE REPRESENTATION FOR COMPUTER CONSULTATIONS

by

Janice S. Aikins

ABSTRACT

This thesis presents a system called CENTAUR, which demonstrates the effectiveness of representing prototypical knowledge in a combination of frames and production rules for performing computer consultations. Key knowledge representation and control structure problems in *production rule* systems similar to MYCIN are identified, and a set of important characteristics of the structures used for representing problem-solving knowledge is given. CENTAUR's frames, or prototypes, complement the production rules to satisfy these characteristics and represent expected patterns of data that permit a more focused, hypothesis-directed approach to problem solving.

Among the characteristics identified as desirable in the representation structures are the ability to *explicitly* represent (a) prototypical cases, (b) the *context* in which knowledge is applied, and (c) the *strategies* for applying that knowledge. CENTAUR's prototypes consist of patterns of knowledge in the domain which serve as broad contexts, guiding the more detailed processing of the production rules. Strategies for the consultation, or control knowledge, are represented in the prototypes separately from other kinds of domain knowledge. This allows the domain expert to specify control knowledge that is specific to each prototype. Examples are presented which demonstrate how this explicit representation facilitates explanations of the system's reasoning. Further, the organization of knowledge in CENTAUR provides a useful framework for acquiring new knowledge.

CENTAUR has been applied to the domain of pulmonary (lung) physiology in which it provides interpretations of pulmonary function tests. The prototypes represent standard pulmonary disease patterns, and the production rules serve as a stylized

form of *attached procedure*. At the highest level, the stages of the consultation itself are represented in a *Consultation* prototype. Thus the advantages of explicit representation apply to control of the consultation process as well.

Other important features of the representation include the use of prototypes as a standard of comparison in order to detect inconsistent or erroneous data, and the representation in production rules of domain expertise to deal with data discrepancies and diagnosis refinement.

Several experiments demonstrating the flexibility of the representation have also been performed. These include the implementation of different top-level prototype selection strategies (*confirmation*, *elimination*, and *fixed-order*), and the use of a second high-level prototype which can review knowledge stored in the domain-level prototypes.

This thesis was submitted to the Department of Computer Science and the Committee on Graduate Studies of Stanford University in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was supported by the Advanced Research Projects Agency under contract MDA 903-77-C-0322. Computer facilities were provided by the SUMEX-AIM facility at Stanford University under National Institutes of Health grant RR-00785-07. The author was supported by the Xerox Corporation under the direction of the Xerox Palo Alto Research Center.

Acknowledgments

It has been both a pleasure and privilege to have worked for the past several years with a group of researchers who have taught me much more than can be reflected within this thesis.

Professor Bruce Buchanan, my principal advisor, has taught me, perhaps most clearly by his own example, what it means to do research. I am indebted to him for both his time and wisdom.

Professor Ed Feigenbaum played a major role in the early stages of my work on the PUFF project, but it is for imparting to me some of his expertise about the "real world" that I am particularly grateful.

The many discussions with my reading committee, Dr. Danny Bobrow, Professor Doug Lenat, and Dr. Nils Nilsson, not only contributed to the body of this thesis, but also broadened my own perspective on AI.

Years of team effort have gone into both the MYCIN and PUFF projects, and it was from my joint role in both of these projects that the foundations for the ideas expressed in this thesis arose. I am grateful to the members of both research teams: the PUFF researchers, Larry Fagan, John Kunz, Dianne McClung, and Penny Nii; Carli Scott and Bill vanMelle, who provided invaluable assistance on questions dealing with MYCIN and EMYCIN; Jim Bennett, who not only read the first complete draft of this thesis, but who, with Avron Barr was always available to discuss a new idea; the medical members of the MYCIN gang, Bob Blum, Greg Cooper, Ted Shortliffe, and Victor Yu, who helped to make CENTAUR accurate, as well as attractive to the physician user; and to Dr. Bob Fallat, because success in creating expert problem solvers depends to a great extent on the cooperation of human experts, and he is one of the best.

I am indebted to the staffs at both the SUMEX-AIM computer facility, where this research was carried out, and the Stanford Artificial Intelligence Laboratory where this document was prepared.

I also wish to thank Jack Lawrence and the many helpful people at the Xerox Palo Alto Research Center who have sponsored my Xerox Corporation Fellowship over the past several years.

Finally, a very special thanks

To my dear friends, Bill Clancey, Randy Davis, Elaine Kant, and Reid Smith, who were always there to give needed advice;

To my parents for their love and support;

To Doug, who went far beyond the call of husbandly duty to greatly improve the stylistic quality of this thesis, who was a constant source of encouragement during the difficult times, and whose faith in me was an essential motivation throughout.

Table of Contents

Appendix Page

Chapter 1

Introduction

1.1 Setting and Themes

What knowledge must be available to a computer system in order for it to achieve expert performance in some domain? How should that knowledge be represented? It is the contention of this thesis that representations of prototypical situations encountered in a domain are one type of knowledge critical to the performance and explanation capabilities of an expert system. The focus of this research has been *computer consultation* systems, that is, systems which interact with users to perform difficult real-world tasks. This thesis presents a consultation system, called **CENTAUR**,[1] that uses knowledge about prototypical situations to guide its processing and to explain its performance to the system user. CENTAUR represents its knowledge as a combination of frames and production rules and performs tasks in the domain of pulmonary (lung) physiology. The frames are called **Prototypes** because they represent typical situations which can be used as a basis for comparison to the actual situation given by the data.[2]

Much of Artificial Intelligence research has focused on determining the

[1] A Centaur is a creature of Greek mythology having the head, trunk, and arms of a man, and the body and legs of a horse. It has the intelligence of man and the strength of a horse, thus combining some of the virtues of each. The CENTAUR system similarly combines the best qualities of production rules and frames in its knowledge representation.

[2] The term prototype has been given the same meaning by other researchers. For example in KRL [Bobrow and Winograd, 1977], a prototype is a special kind of unit representing a hypothetical individual that is the typical member of a class. In [Brachman, 1978] a prototype is a "generalized individual".

appropriate knowledge representations to use in order to achieve high performance from knowledge-based systems. Systems using production rules (e.g., [Shortliffe, 1976], [Buchanan and Feigenbaum, 1978]), and frames (e.g., [Pauker and Szolovits, 1977], [Goldstein and Roberts, 1977]), have been advocated by other researchers. Recently, combinations of knowledge representations have been tried (e.g., [Lenat, 1976]), as researchers are becoming increasingly aware of the need for more representational power in working with real-world domains. A central theme of this research is that it is not the *kind* of knowledge structure that is critical, but that the chosen structure(s) must be *expressive* enough to represent a variety of types of knowledge *explicitly*; that is, the system should have direct, manipulatory access to the knowledge as opposed to having the knowledge "built-in". For example, the function or purpose of the knowledge in the system (e.g., for guiding reasoning or for inferring new information), and the context[3] in which the knowledge is applied, should be explicit.

CENTAUR's combination of prototypes and rules has resulted in a knowledge representation that is expressive enough to represent the wide variety of knowledge necessary in performing pulmonary function interpretations. The prototypes provide the explicit context which guides the more fine-grained knowledge of the production rules. The rules are grouped according to their function in the consultation and are attached to slots in the prototype. Typical patterns of data are represented by each prototype, allowing detection of inconsistent or erroneous data. The overall control structure is sensitive to the initial data, and to

[3] The context as used here refers to the set of facts or preconditions which when taken together describe the situation in which the knowledge is applicable.

the prototype that is being explored, which results in a more focused consultation. Further, the explicit representation of knowledge has greatly facilitated other uses of the knowledge base, such as for explanation of system performance and acquisition of new knowledge.

There has been an increasing emphasis in AI on systems that not only perform well, obtaining a solution as quickly as possible, but that also give acceptable explanations of their performance. (See, for example, [Davis, 1976] and [Swartout, 1977].) In cases in which the explanation system uses the same knowledge structures used by the performance system, even more stringent requirements are placed on the chosen representation. Representations that serve the performance system well and get correct results may be inappropriate as explanations of system *performance*. For example, a production rule may express knowledge at a very detailed level that misses the basic principle behind the rule. Prototype explanations in CENTAUR provide a broad context in which to view the more detailed rule explanations. The explicit representation of the purpose of the knowledge in the *performance system also* has helped the system to justify its conclusions clearly for the user. Thus the knowledge representation in CENTAUR handles both the explanation and performance tasks well.

Knowledge acquisition systems (e.g., [Davis, 1976]) have placed new requirements on the knowledge representation, making it necessary, for example, to allow easy inspection of the knowledge base by experts seeking to add to the breadth of knowledge represented, or to modify the existing knowledge. Again, the explicitness and clarity with which the knowledge is represented is as critically important as the overall organization of the knowledge.

Another trend in AI research is to allow multiple uses of the same knowledge base. In addition to those performing medical consultations, for example, a system has been constructed to teach medical expertise ([Clancey, 1979]). Many of these large knowledge bases represent a substantial investment in the time both of the experts whose expertise is represented and of the computer scientists who work *with them to encode the* expertise into the chosen representations. The payoff in using the knowledge base for more than one task is significant. The chosen knowledge representation thus should not be so tied to the task that multiple uses are awkward or even impossible. One method used to accomplish this is to separate the control structure *within the system from the inference knowledge*, so that control can be modified without necessitating changes to the inference knowledge.

In CENTAUR, control knowledge is represented within each prototype, allowing *context-specific* control, and separating control knowledge from other knowledge in the system. Thus the expert can specify "what to do" in a given context, as an important part of the knowledge about the domain that is distinct from the inferential knowledge used in the consultation. Further, explicit representation of control knowledge allows CENTAUR to provide explanations of control processes.

At the highest level in CENTAUR, the "typical consultation" is represented as a prototype with the various stages of the consultation listed in control slots. Not only does this explicitly define the consultation's control process, but it also allows the flexibility of adding or omitting a stage, and of *more easily experimenting with control* modifications.

A *second task*, *that of reviewing* knowledge stored in the prototypes, is also

represented by a prototype. The REVIEW task allows the user to specify one of the prototypes, and then reviews for him the "typical" features expected in that prototype, as well as any control knowledge associated with the prototype. The REVIEW task illustrates the multiple uses of CENTAUR's domain-level prototypes for a task that focuses on the advantages of the representation for explanation.

1.2 The Task and Domain

The task chosen for this research is that of interpreting measurements from pulmonary (lung) function tests administered to patients in a pulmonary function laboratory. The tests used here were performed in the pulmonary function laboratory at Pacific Medical Center in San Francisco. The laboratory includes several pieces of equipment designed to measure the amount of gas in the lungs and the rates of flow of gases into and out of the lungs. The job of the pulmonary physiologist is to interpret these measurements to determine the presence and severity of lung disease in the patient. An actual set of measurements and interpretation statements is shown in Figure 1.1. The test measurements listed in the top half of the figure are collected by the laboratory equipment. The pulmonary physiologist then dictates the interpretation statements to be included in a manually-prepared report. One sample measurement is the patient's *Total Lung Capacity (TLC)*, that is, the volume of air in the lungs at maximum inspiration, shown on the fourth line of measurements. This patient's TLC (as a percent of the predicted TLC) is 129. All of the measurements are given as a percent of the predicted values for a normal patient of the same sex, height, and weight, so that a value of 100 for the TLC would be the predicted normal value. Therefore, the TLC for this patient is high, indicating the presence of

Obstructive Airways Disease. The interpretation and final diagnosis is a summary of this kind of reasoning about the combinations of measurements taken in the lung tests.

```
PRESBYTERIAN HOSPITAL OF PMC
 CLAY AND BUCHANAN, BOX 7999
 SAN FRANCISCO, CA. 94120
  PULMONARY FUNCTION LAB

WT 40.8 KG,  HT 161 CM,  AGE 65   SEX F
REFERRAL DX-
*********************************************************TEST DATE 5-13-76
                                 PREDICTED                           POST DILATION
                                  (+/-SD)     OBSER(%PRED)          OBSER(%PRED)
INSPIR VITAL CAP (IVC)  L         2.7(0.6)     2.3   ( 83)
RESIDUAL VOL      (RV)  L         2.0(0.1)     3.8   (193)            3.1   (154)
FUNC RESID CAP   (FRC)  L         2.9(0.3)     4.6   (158)            3.9   (136)
TOTAL LUNG CAP   (TLC)  L         4.7(0.7)     6.1   (129)            5.5   (116)
RV/TLC                  %         42.          62.                    55.

FORCED EXPIR VOL (FEV1) L         2.3(0.5)     1.5   ( 66)            1.6   ( 71)
FORCED VITAL CAP (FVC)  L         2.7(0.6) -   2.3   ( 85)            2.4   ( 88)
FEV1/FVC                %         82.          64.                    66.
FORCE EXP FLOW 200-1200L/S        3.6(0.8)     1.8                    1.9
FORCED EXP FLOW 25-75% L/S        2.6(0.5)     0.7                    0.7
FORCED INS FLOW 200-1200L/S       2.5(0.5)     2.5                    3.4

AIRWAY RESISTANCE(RAW) (TLC= 6.1)  2.5         1.5                    2.2
DF CAP-HGB=14.5 (DSBCO)(TLC= 4.8)  23.        17.4   ( 72)
***************************************************************************
INTERPRETATION

The vital CAPACITY is low, the residual volume is high as is the total lung
capacity, indicating air trapping and overinflation.  This is consistent
with a moderately severe degree of airway obstruction as indicatEd by the
low FEV1, low peak flow rates and curvature to the flow volume loop.
Following isoproterenol aerosol there is virtually no change.

The diffusing capacity is low indicating loss of alveolar capillary
surface.

Conclusion:  Overinflation, fixed airway obstruction and low diffusing
capacity would all indicate moderately severe obstructive airway disease of
the emphysematous type.  Although there is no response to bronchodilators
on this one occasion, more prolonged use may prove to be more helpful.

PULMONARY FUNCTION DIAGNOSIS:  OBSTRUCTIVE AIRWAY DISEASE, MODERATELY
                               SEVERE, EMPHYSEMATOUS TYPE
```

FIGURE 1.1 Verbatim Copy of Pulmonary Function Report Dictated by Physician

CENTAUR's principal task is to interpret such a set of pulmonary function test results, producing a set of interpretation statements and a *diagnosis for each* patient. The domain of pulmonary physiology was chosen for several reasons. First, the interpretation of *pulmonary function* tests is *a* problem that occurs daily in hospitals around the world, so a consultation system that captures the expertise *involved in interpreting* these tests and can give assistance in providing these interpretations fulfills a practical need. Second, the amount of domain-specific knowledge involved in pulmonary function testing is small enough to make it feasible for a single researcher to acquire, understand, and represent that knowledge. Third, the domain of pulmonary physiology is a fairly insular field of medicine that does not require representing other large bodies of knowledge on other diseases in order to *produce accurate diagnoses* of pulmonary disease in the patient.[4]

1.3 A Brief History of the Project

This research developed from work done on the MYCIN system [Shortliffe, 1976], which uses a knowledge base of production rules to *perform* infectious disease consultations. Initially, a MYCIN-like production rule system called PUFF [Kunz, et al., 1978] was written to perform pulmonary *function test interpretations*. PUFF was built using a generalization of the MYCIN system called EMYCIN [vanMelle, 1980]. EMYCIN, or "Essential MYCIN", *consists of the domain-independent* features of MYCIN, principally the *rule* interpreter, explanation, and knowledge acquisition

[4] This was a problem in the MYCIN system which determined a diagnosis and therapy for infectious disease cases. *Often, the results produced by* the system suffered because it lacked knowledge about related diseases that were also present in the patient.

modules. It provides a mechanism for *representing* domain-specific knowledge in the *form of production rules*, and performing consultations in that domain. Just as MYCIN is EMYCIN plus a set of facts and rules about the diagnosis and therapy of infectious diseases, PUFF is EMYCIN plus a pulmonary disease knowledge base.

PUFF was written in INTERLISP [Teitelman, 1978] and runs on the DEC system PDP-10 at the Stanford SUMEX-AIM computer facility. In order to *run* the *PUFF* system on the PDP-11 at Pacific Medical Center, a second version of PUFF was created by translating the production rules into *procedures and writing them* in BASIC. That system is currently being used in the pulmonary function laboratory, and provides lung test interpretations for *the approximately* six to eight patients examined there each day.

The form of the interpretations generated by PUFF is shown in Figure 1.2. This report is for the same patient, seen four years later, as the report in Figure 1.1. The pulmonary function test data are set forth, followed by the interpretation statements and a pulmonary function diagnosis, as was done in the earlier typed report. The pulmonary physiologist checks the PUFF report, and when it is required, requests that a typist make changes in the interpretation or diagnosis statements. Approximately 95% of the reports that PUFF generates are accepted without modifications. The change made to most of the rest simply adds a statement to compare the interpretation of the lung tests with tests taken during previous visits. *For example*, statements such as *"These test results are consistent with those of previous visits"* or *"These test results show considerable improvement over those in the previous visit" might be made*. PUFF (and CENTAUR) do not represent knowledge about multiple visits, so statements such as these must be added.[5]

[5] This is not a limitation of the formalism, but rather a restriction on the scope of the problem that was being solved by these research efforts.

```
PRESBYTERIAN HOSPITAL OF PMC
 CLAY AND BUCHANAN, BOX 7999
 SAN FRANCISCO, CA. 94120
  PULMONARY FUNCTION LAB

WT 40.8 KG,  HT 161 CM,  AGE 69   SEX F
REFERRAL DX-
*******************************************************TEST DATE 05/13/80
                              PREDICTED                   POST DILATION
                              (+/-SD)    OBSER(%PRED)     OBSER(%PRED)
INSPIR VITAL CAP (IVC)  L     2.7        2.3   ( 86)      2.4   ( 90)
RESIDUAL VOL     (RV)   L     2.0        3.8   (188)      3.0   (148)
TOTAL LUNG CAP   (TLC)  L     4.7        6.1   (130)      5.4   (115)
RV/TLC                  %     43.        62.              56.

FORCED EXPIR VOL (FEV1) L     2.2        1.5   ( 68)      1.6   ( 73)
FORCED VITAL CAP (FVC)  L     2.7        2.3   ( 86)      2.4   ( 90)
FEV1/FVC                %     73.        65.              67.
PEAK EXPIR FLOW  (PEF)  L/S   7.1        1.8   ( 25)      1.9   ( 26)
FORCED EXP FLOW 25-75% L/S    1.8        0.7   ( 39)      0.7   ( 39)
AIRWAY RESIST(RAW) (TLC= 6.1) 0.0(0.0)   1.5              2.2

DF CAP-HGB=14.5   (TLC= 4.8)  24.        17.4  ( 72)  ( 74%IF TLC = 4.7)
*************************************************************************

INTERPRETATION: ELEVATED LUNG VOLUMES INDICATE OVERINFLATION.  IN ADDITION,
THE RV/TLC RATIO IS INCREASED, SUGGESTING A MODERATELY SEVERE DEGREE OF AIR
TRAPPING.  THE FORCED VITAL CAPACITY IS NORMAL.  THE FEV1/FVC RATIO AND
MID-EXPIRATORY FLOW ARE REDUCED AND THE AIRWAY RESISTANCE IS INCREASED,
SUGGESTING  MODERATELY  SEVERE  AIRWAY  OBSTRUCTION.  FOLLOWING
BRONCHODILATION, THE EXPIRED FLOWS SHOW MODERATE IMPROVEMENT.  HOWEVER, THE
RESISTANCE DID NOT IMPROVE.  THE LOW DIFFUSING CAPACITY INDICATES A LOSS OF
ALVEOLAR CAPILLARY SURFACE, WHICH IS MILD.

CONCLUSIONS: THE LOW DIFFUSING CAPACITY, IN COMBINATION WITH OBSTRUCTION
AND A HIGH TOTAL LUNG CAPACITY IS CONSISTENT WITH A DIAGNOSIS OF EMPHYSEMA.
ALTHOUGH BRONCHODILATORS WERE ONLY SLIGHTLY USEFUL IN THIS ONE CASE,
PROLONGED USE MAY PROVE TO BE BENEFICIAL TO THE PATIENT.

PULMONARY FUNCTION DIAGNOSIS:
1.  MODERATELY SEVERE OBSTRUCTIVE AIRWAYS DISEASE.
      EMPHYSEMATOUS TYPE.
```

FIGURE 1.2 Pulmonary Function Report Generated by PDP-11 Version of PUFF

PUFF serves as a useful tool to the pulmonary physiologist, and thus is a very satisfactory and exciting result of the research done on the production rule consultation systems. Although PUFF's performance results were excellent, there were difficulties in working with the knowledge represented in the system:

- representing prototypical patterns,
- adding or modifying rules to represent additional knowledge,
- altering the order in which information is requested during the consultation, and
- explaining system performance.

These same problems were present in similar rule-based systems, and motivated the creation of CENTAUR. Much of this thesis analyzes the problems that arose in PUFF and explains why the solutions offered by the organization and explicit representation of knowledge provided by CENTAUR's prototypes are important considerations for designing knowledge representations for other knowledge-intensive performance systems.

1.4 Structures for Representing Knowledge

The structures used to represent knowledge in PUFF were IF-THEN, condition-action, or "production rules" of the form shown in Figure 1.3. The "IF" part of the production rule states a set of conditions (the premise clauses) in which the rule is applicable. The action, or "THEN" part of the production rule, states the appropriate conclusions to make when the conditions are satisfied. Two of the advantages

[Davis and King, 1977] of using production rules to represent knowledge are that they are **modular**, so that rules can be added, deleted, or modified without *directly* affecting other rules, and that they are **uniform** in structure with all knowledge being encoded in the same constrained syntax, and can be easily understood by another part of the system in order to examine or modify the rules automatically. (See for *example*, [Davis, 1976] *and* [Waterman, 1970].)

```
CONDITION
          If 1) -----, and
             2) -----, and
             3) -----
ACTION
          Then  -----.
```

FIGURE 1.3 Sketch of a Production Rule

These characteristics of modularity and uniformity have also caused problems in rule-based systems. There are implicit groupings of rules that apply in specific *situations and at certain stages of the consultation*, but there is no explicit indexing of these rules by situations and by stages, like that depicted in Figure 1.4. The figure shows a grid superimposed over a knowledge base of rules, grouping rules by situations and stages in which they are applied, and indicating in a graphic sense an organization of rules such as is provided in CENTAUR.

In rule-based systems the modularity of the rules prevents organization of the knowledge base in a way that would identify groupings of similar rules and would be useful in making modifications to sets of rules or in identifying interactions between rules. Adding or modifying rules may have *indirect* effects on other rules that are

difficult to predict without these explicit groupings. The uniformity of structure often forces different types of knowledge to be represented using the same syntax, and therefore hides the function of the knowledge in the system. For example, rules that are written to control the invocation of other rules, to set default values, or to summarize data, are not distinguishable from rules used to infer new information.

	SITUATION 1	SITUATION 2	SITUATION 3	. . .
STAGE 1	R R R R	R R R	R R R	
STAGE 2	R R R R R	R R R	R R	
STAGE 3	R R	R R R R	R R R R R R R	
⋮				

FIGURE 1.4 Organization of Rules by Situations and Stages

Fortunately, many of these problems are handled well by the mechanism of the **frame** [Minsky, 1975], as sketched in Figure 1.5. A frame is a structure that ties together knowledge about a given situation, and provides expectations about what objects will be present in the situation and what events will occur in the situation. The frame is composed of a set of **slots** and **values** that *specify* the expected objects or events. The slots provide an explicit "place" for information in the frame, so that missing information is evident, and the system can better judge when enough information is known to determine a solution for the problem. Frames thus provide a sense of how *complete* the solution to the problem is, which is unavailable in the rule

systems. Frame systems allow classification of new situations in terms of the stored frame situations. The system attempts to fill in the slots of each frame with values, in order to determine whether there is a match of the *expectations specified by* the slots in the frame with the new situation.

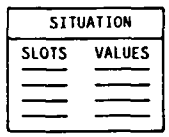

FIGURE 1.5 Sketch of a Frame

CENTAUR's prototypes *are one type of frame, designed* to complement production rules and to solve many of the representation and control problems evident in the rule-based systems. Rules are one type of value for slots in a prototype. *This association of rules with prototype* slots *organizes* the CENTAUR knowledge base into groups of rules as illustrated in figure 1.4. The prototypes are the explicit situation in which the rules are applied. Rules are organized according to stages in which *they are relevant, and each group of these rules* is *the* value of a slot representing knowledge to be applied during that consultation stage.

Each pulmonary disease prototype in CENTAUR represents expected lung test results for one pulmonary disease or its subtype. The characterizing lung tests are represented as slots of the prototype, and are themselves frame-like structures, called **Components**. That is, components are slots whose values are represented as

separate frames. This relationship is diagrammed in Figure 1.6 below. The components are used to represent other characterizing features of each pulmonary disease, *such as an expected referral* diagnosis or whether the patient has a sputum-producing cough. The overall goal of the system is to match the actual test results and patient data with one or more of the prototypes. The immediate goal of *the system, in terms of what information* is most critical to find out next, is determined by the prototype that is being explored, and the component slots that are defined for that prototype.

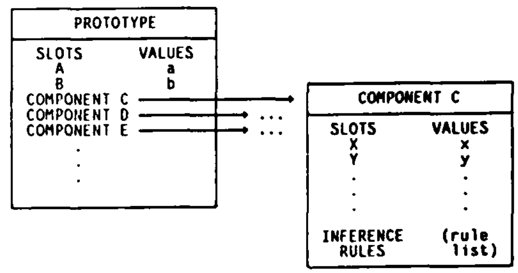

FIGURE 1.6 The Relationship between Prototypes and Components

One of the slots (called *"INFERENCE RULES"*) in the component frame contains a set of production rules used to infer a value for the component. Questions are asked of the user when there are no rules associated with the component, or when the rules fail to infer a value. These production rules are a form of procedural attachment with a constrained, stylized syntax, which makes them easier to examine

than general procedures. This constrained syntax affords many advantages, such as ease of acquisition and modifiability as discussed in [Davis and King, 1977]. Prototypes guide the invocation of the production rules, focusing the search for new information and eliciting the most relevant information from the user.

The prototypes provide a structure for representing knowledge not represented in rule-based systems: **expected patterns of data**. They also provide an explicit means of representing some of the knowledge that was represented implicitly in the rules, including control knowledge and default values. The two boxes in Figure 1.7 illustrate which structures are used to represent domain knowledge in each system. As is shown, CENTAUR represents some new knowledge in prototypes, as well as some knowledge that had been in rules, and retains some of the PUFF rules as attached procedures.

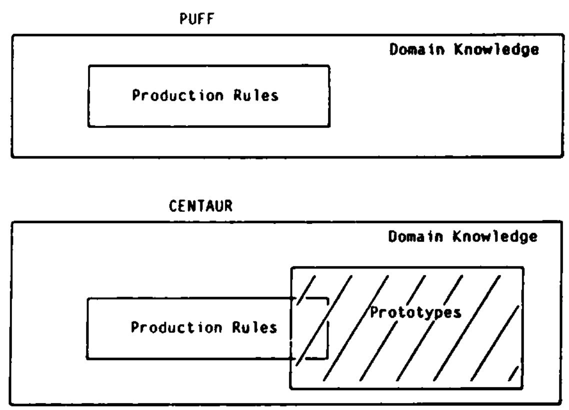

FIGURE 1.7 Representation of Knowledge in PUFF and CENTAUR

1.5 Applicability of the Formalism

CENTAUR's mix of production rules and prototypes has worked well for the pulmonary function interpretation problem, but what type of problems can it handle in general? The approach being used in CENTAUR is one of *hypothesize and match* as described in [Newell, 1973], that is, an attempt is made to match representations of classes of hypotheses against the actual data in the case. Diagnostic problems, of which the pulmonary function problem is one example, are handled well by this methodology. In CENTAUR's terminology, *prototypes* represent classes of hypotheses. Attempts to instantiate prototypes to determine whether there is a match to known data may require new information to be determined, resulting in questions being asked of the user.

The chosen domain must be one in which prototypical situations can be identified and represented as hypothesis classes. It is also important that there be some natural hierarchy of the hypotheses so that only a small subset are being considered at any one time. It is helpful, although not essential to the methodology, that some standard set of initial data be available to suggest hypothesis classes that are most likely to match. If these initial data are not available, CENTAUR will default to testing all possible hypothesis classes.

1.6 A Second Generation System

CENTAUR represents one of very few "second-generation" systems in the field of AI. Others include HSP [McCracken, 1979], a system using a production system architecture for a version of the Hearsay-II speech understanding system, and the

various versions of GPS [Ernst and Newell, 1969]. Second-generation systems are those that perform the same task already accomplished by another system, but do so using some variation in representation of knowledge or control structure, or both. One goal of second-generation systems is experimentation: to demonstrate by doing the same task done by the first-generation system whether the altered knowledge representation or control structure results in *improved performance*. For CENTAUR the specific goal was to demonstrate the advantages of using prototypical knowledge to guide system processing and to explain system performance. CENTAUR's performance was improved over PUFF's because consultations were more focused with fewer questions being asked of the user. Further, as is documented in Section 8.5 CENTAUR's diagnoses agreed more frequently with human physicians than did PUFF's. CENTAUR's organization of knowledge around prototypical cases also allowed *improved explanations and easier knowledge acquisition.*

1.7 Organization of This Report

This thesis presents, in Chapter 2, a brief background description and example of the PUFF system for performing pulmonary function interpretations. Chapter 3 gives a more detailed study of the problems that motivated this research. Chapter 4 presents a CENTAUR consultation using the same patient case as is used for the PUFF example in Chapter 2. Chapter 5 describes the prototypes and production rules that comprise CENTAUR's knowledge representation. Chapter 6 analyzes the control structure as it is represented in the prototypes, and as it unfolds during the various stages of the consultation. Chapter 7 presents samples of explanation and knowledge acquisition routines in CENTAUR and also describes an additional task: the

review of knowledge contained in the prototypes. Finally, Chapter 8 summarizes the main contributions of this research, and presents the performance results of testing the CENTAUR system. A glossary of medical terms used in this report is included in Appendix A for reference. A list of the general control tasks used in CENTAUR's operation is given in Appendix B. Appendix C specifies the templates for the three frame-like data structures used in CENTAUR: prototypes, components, and facts. Complete Lisp and English versions of the Obstructive Airways Disease prototype are presented in Appendix D.

Chapter 2

Background--The PUFF system

2.1 Introduction

CENTAUR was created in response to problems encountered in working with the knowledge representation and control structure of purely rule-based systems. In order to better understand the nature of these problems, this chapter presents a brief background description and example of one of these systems, PUFF, which performs consultations in the domain of pulmonary physiology.

2.2 PUFF Knowledge Representation

The knowledge base of the PUFF system consists of (a) a set of 64 *production rules* dealing with the interpretation of pulmonary function tests and (b) a set of 59 *clinical parameters*. The clinical parameters in PUFF represent pulmonary function test results, for example *TOTAL LUNG CAPACITY* and *RESIDUAL VOLUME*; patient data, for example *AGE* and *REFERRAL DIAGNOSIS*, and data which are derived from the rules, such as *FINDINGS* associated with a disease and *SUBTYPES* associated with the disease. There may be auxiliary information associated with the clinical parameters, such as a list of expected values and an English translation used in communicating with the user.

The production rules operate on associative *(attribute-object-value)* triples, where the attributes are the clinical parameters, the object is the patient, and the

values are given by the patient data and lung test results. Questions are asked during the consultation in an attempt to fill in values for the parameters.

2.2.1 Facts

Each associative triple represents a fact about the patient, and has associated with it a measure of the strength of belief in that fact, termed a **Certainty Factor** or **CF**. The Certainty Factor ranges in value from -1 to 1.[1] A CF of 1 indicates total certainty in the fact, while a CF of -1 indicates total certainty in the negation of the fact. The certainty factor mechanism allows the system to deal with inexact or judgmental reasoning, and enables it to handle statements such as, "The high Total Lung Capacity *suggests* (but is not conclusive of) Severe Obstructive Airways Disease." It also allows the *possibility of more than one value for the same parameter*, if the system finds evidence for each one. For example, the different lung test results may indicate differing degrees of severity for a single disease, resulting in multiple values of degree for that disease. Certainty Factors are discussed more fully in Section 2.4.3.

Some samples of PUFF associative triples (and CFs) are shown in Figure 2.1. The LISP code is shown first, followed by an English translation. The third triple illustrates the possibility of multiple values for a single parameter. The fourth triple is an example of using a negative CF to represent the negation of a fact.

---

[1] The internal representation of a CF is from -1000 to 1000. This is done for efficiency reasons in order to use the faster integer arithmetic functions.

(TLCB PATIENT-7 139 1.0)
The Total Lung Capacity of Patient-7 is 139.

(DEG-REV PATIENT-13 INSIGNIFICANT .8)
The degree of reversibility of airway obstruction of Patient-13 is insignificant with a certainty of .8.

(DEG-OAD PATIENT-9 ((MILD .6) (MODERATE .4))
The Degree of Obstructive Airways Disease of Patient-9 is Mild with a certainty of .6 and Moderate with a certainty of .4.

(DILATION PATIENT-60 YES -1.0)
There are no post bronchodilation test results for Patient-60.

FIGURE 2.1 Samples of PUFF Associative Triples

2.2.2 The Production Rules

The production rules consist of one or more premise clauses followed by one or more action clauses. Each premise is a conjunction of predicates operating on associative triples in the knowledge base. The predicates are simple LISP functions. Each premise clause has the following form:

(predicate function object attribute value)

When the premise is true, the clauses in the rule action are executed. Each rule is actually an executable body of LISP code.[2] A sample PUFF production rule is shown in Figure 2.2.

[2] In fact, to test the truth of the premise, the LISP function EVAL is applied (to EVALuate the premise). If the premise clauses are true, EVAL is then applied to the action clauses.


```
RULE011
-------

 If:  1) A: The mmf/mmf-predicted ratio is between 35 and 45, and
         B: The fvc/fvc-predicted ratio is greater than 80, or
      2) A: The mmf/mmf-predicted ratio is between 25 and 35, and
         B: The fvc/fvc-predicted ratio is less than 80
 Then: 1) There is suggestive evidence (.5) that the degree of
          obstructive airways disease as indicated by the MMF
          is moderate, and
       2) It is definite (1.0) that the following is one of the
          findings about the diagnosis of obstructive airways
          disease:  Reduced mid-expiratory flow indicates
          moderate airway obstruction.

 PREMISE: [$AND ($OR ($AND (BETWEEN* (VAL1 CNTXT MMF) 35 45)
                           (GREATERP* (VAL1 CNTXT FVC) 80))
                     ($AND (BETWEEN* (VAL1 CNTXT MMF) 25 35)
                           (LESSP* (VAL1 CNTXT FVC) 80]
 ACTION:  (DO-ALL (CONCLUDE CNTXT DEG-MMF MODERATE TALLY 500)
                  (CONCLUDETEXT CNTXT FINDINGS-OAD
                                (TEXT $MMF/FVC2) TALLY 1000))
```

FIGURE 2.2 A PUFF Production Rule--English and LISP Versions

The rules are written internally in LISP. The user of the system sees the production rules in their English form which is shown first in the figure. The English version is generated automatically from templates, stored with each predicate function, that indicate the roles of the attribute-object-value triples. For example, the template for the predicate BETWEEN* used in the sample rule is described as follows:

Function Template: (BETWEEN* NUM1 NUM2 NUM3)

Translation: "The" NUM1 "is between" NUM2 "and" NUM3

The predicate $AND is the multivalued analogue of a Boolean AND. It returns as its value the minimum of the Certainty Factors of each premise clause.[3] That minimum Certainty Factor becomes the value of the variable TALLY in the action clauses. It is combined (see Section 2.4.3) with the static Certainty Factor given in the action clauses (.5 and 1.0 in the two action clauses of the sample rule) to obtain the final Certainty Factor for each rule conclusion.

The rule syntax allows nested function calls as illustrated by the VAL1 predicate calls nested in the premise clauses. These return the value for a clinical parameter, which is in turn is used to evaluate the outer clause. There may be arbitrarily complex conjunctions or disjunctions of clauses as well.

There are *27* predicates currently used in PUFF rules. In general, they test the value of a parameter to see whether it agrees with a specified value (e.g., Is the Referral Diagnosis Asthma?) or, for numerical parameters, to see whether it is within a specified range of values (as illustrated by the predicates in the sample rule). *There are also predicates* that operate on a scale of symbolic values (such as *none, mild, moderate, moderately-severe,* and *severe* which are used for degrees of disease) to test the position in the scale of a known value or to shift the known value up or down one position in the scale. For example, the function GREATEQDEG tests a value to see if it is greater or equal to a specific degree on the scale, and the function SHIFTUP shifts the known degree up one position on the scale.

[3] Similarly, the $OR is the analogue of a Boolean OR and does a maximization of the CFs of the clauses.

2.3 PUFF Control Structure

The PUFF (EMYCIN) control structure is primarily a goal-directed, backward chaining of production rules. The goal of the system at any time is to determine a value for a given clinical parameter. To deduce a value for that clinical parameter, it tries a pre-computed list of rules whose actions conclude values for the clinical parameter. It generates this pre-computed list automatically for each parameter using the function templates mentioned in the last section. The system scans the rule ACTION and matches it against the templates to determine which parameters are updated by that rule. The rule number is then added to the UPDATED-BY list for each of these parameters.

If in order to evaluate the premise clause of a rule a parameter is needed whose value has not yet been determined, the new system goal then becomes determination of a value for that parameter. This process continues recursively as the system "chains backward" from it's top-level goal to the available data.

If the rules fail to conclude a value for a parameter, a question is then asked of the user in order to obtain that value. An exception to this process occurs for parameters labeled ASKFIRST parameters. These represent information generally known by the user, such as results of pulmonary function tests. For these *parameters it is more efficient simply* to ask a consultation question than to attempt to infer the information by means of rules.

2.4 Other Details of the Representation

2.4.1 Antecedent Rules

In PLANNER terminology [Hewitt, 1972], most of the PUFF rules are *consequent* rules, and are evaluated because of the need to determine information in their actions, or consequents. However, some of the rules are *antecedent* rules, and are evaluated when information needed in their premises, or antecedents, becomes known. A sample PUFF antecedent rule is presented in Figure 2.3.[4]

```
RULE063
-------

 If:  The number of pack-years smoked is known
 Then:  The degree of smoking of the patient is as follows:
        If the number of pack-years smoked is:
        a) less than 1 then: none (1.0),
        b) between 1 and 8 then: mild (1.0),
        c) between 8 and 30 then: moderate (1.0),
        d) between 30 and 60 then: moderately-severe (1.0),
        e) greater or equal to 60 then: severe (1.0);

PREMISE:  ($AND (KNOWN CNTXT SMOKE))
ACTION:  (CONCLUDET CNTXT (VAL1 CNTXT SMOKE)
                 '((LT 1 1000 0 0 0 0)
                   (BT 1 8 0 1000 0 0 0)
                   (BT 8 30 0 0 1000 0 0)
                   (BT 30 60 0 0 0 1000 0)
                   (GE 60 0 0 0 0 1000))
          TALLY DEG-SMOKE '(NONE MILD MODERATE
                      MODERATELY-SEVERE SEVERE))
```

FIGURE 2.3 PUFF Antecedent Rule

[4] This rule is an example of a *tabular rule* in which the rule action is a decision table on one condition, in this case, the value of a single parameter. Tabular rules were created for the EMYCIN systems by Larry Fagan.

Antecedent rules have the same syntax as consequent rules, and differ only in when they are applied. Whenever a value is concluded for a clinical parameter during the consultation, any antecedent rules associated with that parameter are executed. Unlike consequent rules which chain recursively backwards, evaluating antecedent rules will not cause other rules to be tried. That is, if the clause of an antecedent rule refers to a parameter whose value has not yet been determined, the system will *not* try other rules to deduce a value for the parameter. Nothing more will be done with the antecedent rule at that time. If at a later time the needed parameter value is determined by some other means, the antecedent rule will be evaluated again.

The antecedent rules in PUFF are used chiefly for defining one parameter in terms of other known parameters. For example, in the antecedent rule in Figure 2.3, the parameter DEG-SMOKE, which is a symbolic degree of the influence of smoking on the patient's lung disease, is defined in terms of another parameter, SMOKE, a patient datum, which is the number of packs of cigarettes smoked each day multiplied by the number of years the patient has been smoking.

2.4.2 Starting the Consultation

The EMYCIN control structure normally determines values for parameters only *as they are needed by rules* that are being evaluated. However, some parameter values are expected to be known in advance, and the consultation seems more structured when these "initial data" are asked as a group. To allow this, sets of parameters are specified as *INITIALDATA parameters and are associated with an* object in the domain.

PUFF has a single object, the patient, for each of its object-attribute-value triples. When the consultation begins, the INITIALDATA parameters of the patient are the first ones to be determined. In PUFF, these initial data are some of the basic patient data (e.g. *NAME* and *REFERRAL DIAGNOSIS*) and some of the most important pulmonary function test results.

Other parameters may be specified as GOALS which are determined after the INITIALDATA parameters and generally require invoking rules. In PUFF, there is a single GOAL parameter, *INTERPRETATION*, which is used in the action of one of the rules. Determining a value for the INTERPRETATION parameter entails evaluating this "goal" rule. The PUFF goal rule is shown in Figure 2.4. Premise clauses of the rule check for the possibility of each pulmonary disease (and for normal pulmonary function), and determine additional summarizing statements for the interpretation. Evaluation of the premise clauses then sets up sub-goals, causing rules dealing with each disease to be evaluated in turn. The action of the goal rule prints the desired lung function test interpretation and diagnosis of lung disease for the patient.


```
RULE001
-------

        If:  1) An attempt has been made to deduce
                whether there is an interpretation of
                potential obstructive airways disease,
             2) An attempt has been made to deduce
                whether there is an interpretation of
                potential restrictive lung disease,
             3) An attempt has been made to deduce
                whether there is an interpretation of
                a potential diffusion defect,
             4) An attempt has been made to deduce the
                findings about the diagnosis of normal,
             5) An attempt has been made to deduce
                the summary statements about this
                interpretation
        Then:  Print the interpretation of the pulmonary
                  function tests

PREMISE:  ($AND (ONCEKNOWN CNTXT INTERP-OAD T)
                (ONCEKNOWN CNTXT INTERP-RLD T)
                (ONCEKNOWN CNTXT INTERP-DIFDEF T)
                (ONCEKNOWN CNTXT FINDINGS-NORMAL T)
                (ONCEKNOWN CNTXT FINDINGS-SUMMARY T)
ACTION:  (PRINTINTERP INTERPRETATION)
```

FIGURE 2.4 PUFF Goal Rule

2.4.3 Certainty Factors

Certainty Factors are associated with action clauses of rules as a measure of increased belief in the rule conclusion when the premise clauses of the rule are satisfied. For example, the first action clause of the rule in Figure 2.2 states that the evidence is suggestive of the conclusion (.5 out of 1.0), but not completely certain. A Certainty Factor is also associated with each object-attribute-value

triple, or fact, as an indication of the current predominance of confirming (for positive CFs) or disconfirming (for negative CFs) evidence about that fact.

Certainty Factors are not conditional probabilities, but are based on probability theory. The Certainty Factor formalism arose from a desire for a *one-number* calculus. [Shortliffe and Buchanan, 1975] gives a full discussion of the Certainty Factor theory, and it will not be discussed in detail here. The present EMYCIN system uses a modification of the original Certainty Factor combining scheme (as presented in [Shortliffe and Buchanan, 1975]). The EMYCIN function now used to combine two Certainty Factors, X and Y is as follows:

$$F(X,Y) = \begin{cases} X + Y(1 - X), & \text{if X and Y are positive;} \\ \dfrac{X + Y}{1 - \min(|X|,|Y|)} & \text{if } X * Y \text{ is negative;} \\ -F(-X,-Y), & \text{if X and Y are negative;} \end{cases}$$

and, $F(1,-1) = F(-1,1) = 1$.

The original combining function was not sensitive to the *number of items* of confirming or disconfirming evidence, so that a single negative CF of -.9 could cancel out several smaller positive CFs that together were strong confirming evidence for the hypothesis. The new combining function is more stable when combining CFs of unlike sign. (It is identical to the original function for CFs of like sign.) The new function also has the virtue of being commutative, so that a single CF is stored, as opposed to the measures of both confirming and disconfirming evidence that were accumulated separately and then combined, as in the original scheme.

2.5 PUFF Example

The following is an example of an interpretation of lung function tests by PUFF. *Information supplied by the user is preceded by a* **double asterisk (**), and is printed in BOLDFACE.** Annotations are placed in parentheses and written in *italics*. In this example, the patient is determined to have a case of ***Severe Obstructive Airways Disease*** (OAD) of *Asthmatic* type. This is an actual case from the patient files of the pulmonary function laboratory at Pacific Medical Center in San Francisco. However, the patient's identifying number has been changed. This case was chosen because it is a *fairly* typical case of OAD, and thus is representative of many cases in our patient library. The more inquisitive reader may refer to the glossary in Appendix A for definitions of the technical terms.

```
19-Nov-79 14:41:28
[consultation of 23-Feb-77 11:55AM]

*PUFF*
```

(PUFF asks the user for pulmonary function test results and patient data as it executes rules that make conclusions about the interpretation of the tests.)

```
--------PATIENT-7--------
1) Patient's identifying number:
** 7446
2) referral diagnosis:
** ASTHMA
```

(RV, TLC, FVC, etc. are the lung test measurements.)

```
3) RV/RV-predicted:
** 261
4) TLC(body box)observed/predicted:
** 139
```

```
5) TLC(DLCO)observed/predicted:
** 111
6) FVC/FVC-predicted:
** 81
7) FEV1/FVC ratio:
** 40
8) The slope F5025:
** 9
9) The FEV1/FVC threshold:
** 77
10) MMF/MMF-predicted:
** 12
11) the DLCO/DLCO-predicted:
** 117
12) Change in FEV1 (after dilation):
** 31
13) The change in resistance (after dilation):
** 2.0
14) The severity of coughing:
** NONE
```

*(Pack-years = Number of packs per day * Number of years patient has smoked)*

```
15) The number of pack-years of smoking:
** 17
16) The degree of dyspnea:
** NONE
```

(At this point, all of the applicable PUFF rules have been executed and PUFF has completed its questioning of the user.)

****

(The final PUFF interpretation consists of two parts. First, there is a summary of some of the most important pulmonary function data. Second, the English interpretation statements concluded in the rules are printed. The final pulmonary disease diagnosis is printed at the end of the interpretation statements.)


```
SUMMARY OF DATA:

The degree of dyspnea:  NONE
The severity of coughing:  NONE
The number of pack-years of smoking:  17
referral diagnosis:  ASTHMA
RV/RV-predicted:  261
FVC/FVC-predicted:  81
TLC(body box)observed/predicted:  139
FEV1/FVC ratio:  40
MMF/MMF-predicted:  12
the DLCO/DLCO-predicted:  117
The slope F5025:  9

INTERPRETATION:

Obstruction is indicated by curvature of the flow-volume loop.
Forced Vital Capacity is normal but the FEV1/FVC ratio is reduced,
   suggesting airway obstruction.
Low mid-expiratory flow is consistent with severe airway obstruction.
Change in expired flow rates following bronchodilation shows that
   there is reversibility of airway obstruction.
Improvement in expired flow rates following dilation indicates
   reversibility of airway obstruction, and this is confirmed by
   improvement in airway resistance.
Good response to bronchodilators is consistent with an asthmatic
   condition, and their continued use is indicated.
The high diffusing capacity is consistent with asthma.
Elevated lung volumes indicate overinflation.
Airway obstruction is consistent with the patient's smoking history.
The pulmonary diffusing capacity for carbon monoxide is normal.
Obstructive Airways Disease of the asthmatic type.
```

2.6 What the Example Doesn't Show

This PUFF example is *important*, not because of what it shows about the pulmonary function consultation, but because of what it does *not* show. The consultation consists only of a series of questions with user responses, with no *explicit* indication of which *hypothesis* is being tested by the system that prompts each question, and no evidence of major conclusions being made during the consultation. The user of an EMYCIN system can choose a tracing option that will

print the numbers of the rules being tried, and any rule conclusions (with their certainty factors) made during the consultation. (A sample of a portion of this consultation with tracing is given in Figure 2.5.) However, because all of the rules share the same stylized syntax, rules that represent major reasoning steps in the consultation (such as the goal rule in Figure 2.4) are indistinguishable from rules that represent the most detailed inferences. Thus, the tracing mechanism prints *all* rules being tried and *all* rule conclusions.

(The first seven questions are INITIALDATA parameters which are asked of the user without trying rules. This trace begins just as the GOAL parameter, INTERPRETATION has been selected as the current system goal. RULE001, the goal rule, is tried because its conclusion mentions this parameter. In general, the numbers in brackets indicate the depth of the goal form the top-level goal given by [1].)

```
.
.
.

   --[1] Findout: INTERPRETATION of PATIENT-7
Trying RULE001/PATIENT-7;

   --[2] Findout: INTERP-OAD of PATIENT-7
```

(The first premise clause of RULE001 sets up subgoal [2].)

```
Trying RULE002/PATIENT-7;

   --[3] Findout: DEG-OAD of PATIENT-7
```

(Trying RULE002 sets up subgoal [3], and so on.)

```
Trying RULE003/PATIENT-7;

   --[4] Findout: DEG-SLOPE of PATIENT-7
Trying RULE015/PATIENT-7;

   8) The slope F5025:
```

(Trying RULE015 causes a question to be asked because F5025 is an ASKFIRST parameter.)

```
   ** 9
RULE015 succeeded.
Conclude: DEG-SLOPE of PATIENT-7 is SEVERE (.8)
```


(Conclusions from RULE015 are listed.)

```
Conclude: FINDINGS-OAD of PATIENT-7 is $F5025 (1.0)
   --[4] Finished: DEG-SLOPE of PATIENT-7

   --[4] Findout: DEG-FEV1 of PATIENT-7
```

(A new level [4] subgoal is set up, as the system still tries to execute RULE003.)

```
Trying RULE007/PATIENT-7;

   9) The FEV1/FVC threshold:
```

(Another ASKFIRST parameter is used in RULE007.)

```
   ** 77
   .
   .
   .
```

FIGURE 2.5 Sample of Rule Tracing

There is also no indication of the *consultation stages* of gathering initial information, attempting to confirm a hypothesized disease, and determining findings associated with a final diagnosis. The user must take it on faith that the system is progressing through the consultation, intelligently trying various possible diagnoses to interpret a given set of data. It is not evident, for example, that the first seven questions asked of a user are always asked as a standard set of initial data is acquired by the system, that questions eight through fourteen represent the principal inference stage in which the system attempts to confirm the presence and subtype of OAD, and that questions fifteen and sixteen are asked during the final consultation stage in order to determine findings associated with the confirmed OAD diagnosis.

If the production rules are examined as a whole, they fall into groups which are used at various stages of the consultation, but in each individual rule, there is no representation of the stage in which it is applied. Consultation stages are one form of *meta-knowledge*, or knowledge about the knowledge in the domain, that is not represented in production rules. Therefore, even printing out individual rules will not provide this information.

Although the questions being asked give some indication of the system's line of reasoning, the motivation for asking each question is not specified. Consultation questions may seem unreasonable or even irrelevant with respect to a given case if the user is not able to detect the system's line of reasoning from the questions asked. The user of an EMYCIN system may use special keywords in response to a system question to find out which rule is being evaluated, and thus caused the question to be asked. This requires, however, running the separate explanation sub-program (to be discussed in Chapter 7) and therefore can be very time-consuming. The explanation sub-program requires asking for a justification in response to *each question*, *rather than having the reasoning process* displayed as an integral part of the consultation.

The form of this consultation reflects some of the problems posed by the knowledge representation and control structure of the PUFF system. For example, the reason that there is no indication of which diseases are being explored when a question is asked is that there is no explicit representation of the *context* (or the disease state) in which the rules are applied. It is not possible to infer from the consultation typescript itself the answers to questions such as *Was OAD being*

considered when question 11 was asked or was It Diffusion Defect? (the question is relevant to both diseases) or *Were other diseases ever considered?* The answers to these questions are only evident when we look at the detailed trace showing each *rule that was tried during the consultation*. Even then we must abstract from the rule the disease state in which it was applied, in order to determine what disease was being explored when a particular question was asked. This is not an adequate solution for most users. It is interesting to note that in this consultation, PUFF actually explored the possibility of four different disease states and executed rules *for each one*. Some of them, for example Restrictive Lung Disease, required no additional information from the user, so no questions were asked. Therefore, in this case, exploring the possibility of Restrictive Lung Disease represents a significant reasoning step by the system that is completely hidden from the user.

The form of the final interpretation also seems incomplete, as there is little indication of what data produced each statement, and no indication of what data are consistent (or inconsistent) with a final pulmonary disease diagnosis. The format of this interpretation basically presents input data and final conclusions, and forces the user to draw connections between the two.

The incompleteness of PUFF's final data interpretation reflects the need for *additional representational power in its data base*. For example, the reason that there is no indication of which data are consistent with the *final diagnosis* is that there is no representation of *prototypical* data patterns for each disease. For the same reason, there is no indication of data that are inconsistent with the final diagnosis, or of data that are not consistent with *any* disease pattern. *Information*

that can not be accounted for by expected disease patterns may indicate a data error or a gap in the system's knowledge of the domain. Knowledge of such discrepancies can be helpful, not only in indicating possibly erroneous information, but also in guiding the system to explore possible reasons for such discrepancies. Because PUFF does not represent expected disease patterns, it can not utilize this reasoning strategy.

Solutions for handling each of these problems by augmenting the rule form of representation probably are attainable. However, after a careful analysis of the essential capabilities of representation structures used in performing computer consultations, the prototype representation was designed for CENTAUR to serve as a single, complete solution to the problems noted above. The next chapter takes a more detailed look at the problems with the knowledge representation and control structure of the rule-based systems that motivated the design of the prototypes.

Chapter 3

Motivation

3.1 Introduction

In order to understand the reasons why prototypes were added to the knowledge base, it is necessary to understand the problems that occurred while using the rule representation to perform consultations in PUFF and MYCIN. These problems can be grouped roughly into four areas: problems with the knowledge representation itself, problems encountered in the process of acquiring new knowledge or modifying existing knowledge, problems with the control structure of the system, and finally, problems in explaining system performance. This chapter explores each of these areas. Later chapters describe how CENTAUR's representation of knowledge and control structure avoid these problems.

3.2 *Representation of Knowledge*

3.2.1 The Nature of the Rule Knowledge

In PUFF and MYCIN, rules represent a single, *uniform "grain size"* of knowledge that is applied in precise contexts defined by the premise clauses of each rule. The knowledge bases lack any representation of broader, general contexts representing coarse-grained knowledge in which several rules could apply. These general contexts are desirable in order to allow a control structure that would first determine the more general context before searching for detailed information. They could be

used to organize the knowledge base into sets of general and specific knowledge about each topic. Such an organization would make it easier to view the contents of a knowledge base, facilitating knowledge acquisition (as the expert views the knowledge base in order to modify its contents) and explanations of system performance (as the system views its own knowledge base to form explanations).

Rules also represent *highly-specialized knowledge*. One of the primary goals of the MYCIN research was to achieve a system that performs at the level of an infectious disease expert. It was expected that the system would be used as an aid to the physician who needed assistance on a difficult infectious disease consultation. Thus the knowledge in many of the rules deals with difficult or rare cases. As has been noted in [Clancey, 1979], some MYCIN rules have even been *optimized* so that intermediate concepts are not represented. For example, if *X causes Y* and *Y causes Z*, then we can simply represent this fact as *X causes Z*. The rule is then more efficient, but significant reasoning steps are missing because the intermediate clauses are not represented

Similarly, in PUFF, the goal was to produce an interpretation of pulmonary function tests that would be equivalent to the interpretation provided by a pulmonary disease expert. Rules were written to achieve this end, and again, some very basic knowledge about pulmonary disease types was not explicitly represented. For example, no expected sets of values for pulmonary function tests of patients with each type of pulmonary disease were represented. Sets of expected values are necessary for detecting inconsistent information as well as for providing useful knowledge about the domain.

Optimized knowledge occurs in PUFF in the form of clinical parameters defined as combinations of *raw* data measurements. For example, the parameter ***NMD*** is a function of three of the pulmonary function test measurements important in detecting Neuromuscular Disease. PUFF rules use this single derived measurement as a general indicator for Neuromuscular Disease. The rules then refer to one measurement rather than to three. However, NMD is not a standard lung function measurement, and therefore the rules that use it are only understood by those familiar with the system.

Although representation of this basic knowledge is not necessarily important for routine system performance, it becomes critical when the knowledge represented in the system is itself a contribution of the research, as in teaching systems, or when the system attempts to explain its own reasoning. In such cases, the system is left without any explicit representation for this basic knowledge, making it more difficult for the user to examine and understand the knowledge base.

3.2.2 Context Implicit in Rules

Many of the rules in EMYCIN systems represent knowledge implicitly. The context in which each rule applies is often set in some of the premise clauses, with the remaining premise clauses representing actual pieces of medical *expertise*. This implicit context only becomes evident when the rules are viewed as a group. For example, the top half of Figure 3.1 depicts a set of three rules that are applicable when their common premise clauses, A and B, are true. If any one of the rules is viewed in isolation, it is no longer clear that the A and B clauses form a context in which the third clause is tested to form the conclusion. The alternative to this

implicit context representation is to associate all three rules *explicitly* with an AB context (and include in each rule a pointer to that context), as illustrated by the second half of the figure. The approach used in CENTAUR is to associate the rules explicitly with the prototype defining the context in which they are applied.

```
Rule 1:  If A and B and C, then X
Rule 2:  If A and B and D, then Y      implicit AB context
Rule 3:  If A and B and E, then Z

+-------------------------------+
| CONTEXT:  A and B are true    |
|                               |
| Rule 1:  If C, then X         |     explicit AB context
| Rule 2:  If D, then Y         |
| Rule 3:  If E, then Z         |
+-------------------------------+
```

FIGURE 3.1 Implicit versus Explicit Contexts

An actual rule sample from PUFF illustrating the implicit "normal patient" context is shown in Figure 3.2. The first three premise clauses check for the absence of disease, and the information tested in the fourth premise clause actually determines what summary statement will be printed. Not only does this form of organization fail to indicate that this is PUFF's definition of a normal patient, which would be necessary in attempting to explain the rule, but it also requires that a new clause be inserted whenever a new disease is added to the knowledge base stating that the new disease is also absent in a normal patient.[1] CENTAUR's solution, to associate the

[1] In this case we could have introduced an "intermediate parameter" for a "normal patient" and written a new *rule* to *define* "normal patient". The first three clauses of RULE050 then would be replaced by a test to see if the patient was normal. However, this new "context-setting" rule would not be distinguishable from other rules in the system, and adding a new disease to the knowledge base still would entail adding a clause to this rule. Other prototypical knowledge about the normal patient also would be missing.

rule with the NORMAL prototype as the explicit context in which it is applied, is shown in Figure 3.3. Notice that it becomes no longer necessary to alter the definition of a normal patient when a new disease is added to the knowledge base. Other prototypical knowledge about the normal patient in CENTAUR is also clearly defined in the NORMAL prototype for explanatory purposes.

```
RULE050
-------

If:  1) The degree of obstructive airways
        disease of the patient is NONE,
     2) The degree of lung restriction of the
        patient is NONE,
     3) The degree of diffusion defect of the
        patient is NONE, and
     4) The degree of obstructive airways
        disease as indicated by overinflation
        is greater than or equal to mild
Then:  It is definite (1.0) that the following is
          one of the summary statements about this
          interpretation: Pulmonary Function is
          within wide limits of normal.
```

FIGURE 3.2 PUFF Rule--Context Implicit in Premise Clauses

```
RULE050
-------

If:  The degree of obstructive airways disease
        as indicated by overinflation is greater
        than or equal to mild
Then:  It is definite (1.0) that the following is
          one of the summary statements about this
          interpretation: Pulmonary Function is
          within wide limits of normal.

PROTOTYPE:  NORMAL
```

FIGURE 3.3 Explicit Context of CENTAUR Summary Rule

A second, less obvious example of an implicit context in a PUFF rule is illustrated by the rule in Figure 3.4. It would be necessary to see *all* of the PUFF rules in order to determine that there is a particular group of rules, all dealing with cases in which there is a *Diffusion Defect* (as indicated by the second premise clause in the rule), which attempts to relate that diffusion defect to other pulmonary diseases. That is, each rule in this group applies when a diffusion defect has been confirmed. The implicit context is "the patient has a diffusion defect". Looking at this rule in isolation gives no clue that the second premise clause represents this context. In fact, there is no syntactic identification of this grouping of rules in PUFF. The CENTAUR solution, to associate the rules explicitly with the Diffusion Defect prototype, is shown in Figure 3.5.

```
RULE032
-------

  If:  1) The severity of obstructive airways
          disease of the patient is greater than or
          equal to mild,
       2) The degree of diffusion defect of the
          patient is greater than or equal to mild,
          and
       3) The tlc(body box)observed/predicted of the
          patient is greater than or equal to 110
  Then:  It is definite (1.0) that the following is
            one of the conclusion statements about
            this interpretation: The low diffusing
            capacity, in combination with
            obstruction and a high total lung
            capacity would be consistent with a
            diagnosis of emphysema.
```

FIGURE 3.4 Implicit Context of PUFF Rule


```
RULE032
-------

  If:  1) There is evidence for OAD, and
       2) The tlc(body box) of the patient is
          greater than or equal to 110
  Then:  It is definite (1.0) that the following is
            one of the conclusion statements about
            this interpretation: The low diffusing
            capacity, in combination with
            obstruction and a high total lung
            capacity would be consistent with a
            diagnosis of emphysema.

  PROTOTYPE:  DIFFUSION-DEFECT
```

FIGURE 3.5 CENTAUR Refinement Rule Illustrating Explicit Diffusion Defect Context

3.2.3 Control Implicit in Rules

Control knowledge is also represented implicitly in rules by ordering their premise clauses to control the invocation of other rules. Because rules are invoked as new information is needed by the system, control rules can be written with premise clauses referring to parameters whose values are not yet known. This causes the invocation of other rules whose conclusions update those parameters. The order of the premise clauses of a control rule thus determines the order in which the other rules are invoked. For example, the principal PUFF rule that forces *all* of the diseases to be considered in performing the pulmonary function interpretation was shown in Figure 2.4. Invoking this rule causes other rules to be tried in sequence which test for the possibility of each pulmonary disease, form summary

statements, and print a final data interpretation. A similar top-level control rule exists in MYCIN, which causes information necessary to the consultation to be obtained in an order intelligible to physicians, and which then produces a diagnosis and therapy recommendation for the patient.

Other rules that control the order in which questions are asked of the user exist in both MYCIN and PUFF. A subtle example of implicit control is illustrated by the PUFF rule in Figure 3.6 below. This rule invokes other rules in an attempt to determine whether there is Obstructive Airways Disease (Clause One),[2] and if so, to determine the subtype (Clause Two) and findings associated with the disease (Clause Three). If Clause One were inadvertently placed after either Clause Two or Three, the system's questions of the user would probe for more detailed information about Obstructive Airways Disease without having confirmed that that disease was present. For example, by reordering the clauses in RULE002, PUFF might begin its consultation by asking about the patient's smoking history, one of the findings associated with Obstructive Airways Disease, and a question that would be inappropriate in a patient without a smoking-related disease. The order of the clauses is critical here, yet that fact is not explicitly indicated in the rule.

[2] PUFF equates the existence of a disease with the degree of the disease being greater than or equal to mild.


```
RULE002
-------
  If:  1) An attempt has been made to deduce the
          degree of obstructive airways disease
          of the patient,
       2) An attempt has been made to deduce the
          subtype of obstructive airways disease,
          and
       3) An attempt has been made to deduce the
          findings about the diagnosis of
          obstructive airways disease
  Then:  It is definite (1.0) that there is an
           interpretation of potential obstructive
           airways disease
```

FIGURE 3.6 PUFF Rule--Implicit Control

CENTAUR's solution to the representation of control knowledge is to associate control knowledge *explicitly* in slots associated with each prototype, whereby the slot name specifies the point during the consultation when the control is applied. Thus in Figure 3.7, we see CENTAUR's representation of the PUFF rule given above. This control knowledge is represented in two control slots associated with the Obstructive Airways Disease prototype (and defined in Section 5.2.2). It specifies that when OAD is confirmed, the next task is to deduce a degree and a subtype for OAD, and at a later stage in the consultation (when the prototype ACTION slots are executed), to deduce and print findings associated with OAD.

If-Confirmed
Deduce the Degree of OAD
Deduce the Subtype of OAD

Action
Deduce any Findings associated with OAD
Print the Findings associated with OAD

FIGURE 3.7 Explicit Representation of Control Knowledge in OAD Prototype Control Slots

3.2.4 Implicit Functions of Rules

Although most PUFF and MYCIN rules are used to infer new information about the case, some rules have different functions, such as the "control" rules just discussed, or rules that summarize information obtained during the consultation. No new information is deduced in these Summary Rules; their conclusions merely produce summarizing statements. (A sample of such a rule was given in Figure 3.2 above.) Rules are also used to set default values for some parameters whose values could not be determined by other means. Although the rule *format worked well for these* multiple purposes, there was no explicit distinction made between the different types of rules. This frequently led to confusion on the part of the experts working with the knowledge base. For example, it was difficult to determine whether a given rule was written to infer new information, to summarize *information, or to ensure that* other rules would be invoked in an appropriate order.

CENTAUR *groups rules explicitly according to their function* in the consultation. For example, the PUFF rule in Figure 3.2 was classified as a CENTAUR Summary Rule, as shown in Figure 3.3. The rule groups will be discussed fully in Chapter 5. Rules that represented control knowledge were re-represented as prototype slots, as has been discussed. Similarly, component slots in the prototype have as one possible sub-slot a default value, with the result that rules setting default values become unnecessary.

3.2.5 The Ingredients of a Certainty Factor

The reliability of case data, as well as the subjective importance of the data in

determining the diagnosis, are often represented in Certainty Factors associated with rule conclusions in PUFF and MYCIN. Recall that the Certainty Factor is an expert's indication of increased strength of a rule conclusion when all of the premise clauses are true. Often experts will lower a Certainty Factor if they feel that the data being tested in the premise clauses are less reliable; similarly, they may raise a Certainty Factor if the data being tested are more determinative of the diagnosis [3]. In PUFF, *for example*, there are two possible ways of measuring Total Lung Capacity (TLC) of a patient: using a very reliable body plethysmography or "body box" method, and using a less reliable gas dilution method. PUFF represents two parameters, TLCB and TLCD, corresponding to these two measurement methods. Conclusions in PUFF rules that are based on the gas dilution measurement (TLCD) have lower Certainty Factors than those based on the body box measurement (TLCB), although they are all testing a value for the *same* lung capacity measurement.

CENTAUR separates subjective measure of increased belief (expressed as a Certainty Factor) from the concept of reliability, or diagnostic importance of a *measurement*. The reliability of the TLCB and TLCD test measurements are expressed as a number (from 1 to 5), thus separating the reliability of each measurement from its value.

3.2.6 Detecting Atypical or Erroneous Data

There is only a very limited capability in EMYCIN systems for detecting atypical

[3] There is clearly a relationship between reliable data and those that are good diagnostic indicators, and conversely, data that are inherently unreliable and those that are not good diagnostic indicators.

or erroneous data values. Each clinical parameter has associated with it a list of expected values that is checked when a value is supplied for that clinical parameter. The clinical parameters are not specific to any one disease, but apply very generally to all diseases in the domain. Therefore, the lists of expected *values are also very* broad, regardless of how much other information is known in a specific consultation which could narrow the expected possibilities. For example, in PUFF there is a single clinical parameter representing each pulmonary function test and a single list of expected values. In the pulmonary function domain, however, a value for a pulmonary function test result that is within a range of expected values for one disease could indicate erroneous data if a different disease were being considered. Because PUFF has no representation of clinical parameters that are *specific to each disease*, the list of expected values can be very broad. Many of the possible ranges of values for clinical parameters in PUFF are, in fact, so broad that the list of expected values is given the symbolic value, NUM, to indicate only that a number is expected. This same set of expected values is used by the system as a response to users who ask what values are possible answers to a system query. Thus in PUFF, if a user is confused about the possible range of values for a pulmonary function test result, the system can supply no more information than that a numerical value is expected.

In CENTAUR, each prototype lists expected values for the parameter associated with it so that atypical values for the parameter can be pointed out to the user as possibly erroneous data. Further, lists of expected values are *specific to the context* defined by each prototype so that, for example, a more narrow range of numbers can be given when a user asks what values are possible for a particular answer.

3.2.7 Representing Typical Patterns of Data

Also missing in EMYCIN systems is any representation of typical patterns of data expected in a given context which might serve as a standard of comparison for the actual data being considered. Thus the systems lack the ability to single out cases that do not follow consistent patterns, and which might warrant additional consideration by the physician user.

CENTAUR's prototypes represent these *typical patterns* of data, and thus serve as a *standard of comparison* for detecting atypical or erroneous data. Cases are stored with an indication of which prototypes were found to match, facilitating *retrieval of appropriate cases* for testing new or modified rules associated with particular prototypes. This feature has been quite useful during system development, permitting tests of changes made to the knowledge base. Typical cases can also be saved as *examples for educational or explanatory purposes.* In addition, a variety of cases encompassing the system's range of knowledge can be selected and used to illustrate its capabilities.

3.3 Knowledge Acquisition

One reason often cited for using production rules (for example, [Davis and King, 1977] and [Davis, Buchanan, and Shortliffe, 1977]) is that there is no *direct* interaction of one rule with the others, a characteristic which facilitates adding rules to the knowledge base or modifying existing rules. In practice, however, the *rules are actually highly interconnected.* If we want to add or modify a rule, we must first identify the set of rules that could invoke it and also the rules that it invokes in turn,

and then determine whether *changes in these rules also must be made*. *If we modify* one or more of these rules, then the process repeats.

In EMYCIN systems, the only indexing of rules is according to clinical parameters used in their premise and action clauses: each clinical parameter has associated with it a list of the rules that use that clinical parameter in their premise clauses, and a list of the rules that use it in their action clauses. Although this helps to determine the groups of rules that will be affected when modifications are made, the lack of an *explicit context* has complicated knowledge acquisition. For example, when a second infectious disease (meningitis) was added to the original (bacteremia) rules in the MYCIN system, it was necessary first to add a clause *If the infection is bacteremia* to all of the bacteremia rules in order to limit their application to bacteremia cases. Similarly, the first clause of each meningitis rule was *If the infection is meningitis*. Each rule thus required stating an implicit context, and *every* old rule in the system had to be modified in order to accommodate the new set of rules. Further, rules that now apply to both of these infections might not apply to a third infection, thus necessitating the addition of even more context-setting clauses for future expansion of the knowledge base.

As is evident from this example, separating the context in which knowledge is applied from the knowledge itself, as is done in CENTAUR, has important implications for knowledge acquisition. Further, the association of sets of knowledge in CENTAUR within the explicit context of a prototype more clearly identifies the knowledge that could be affected by a change.

The *implicit representation of control knowledge* in EMYCIN systems also has

an adverse affect on knowledge acquisition. *Removing or modifying clauses of a control rule* can alter the system's behavior in unexpected ways, since implicit control knowledge also will be altered. However, changing a rule that summarized or inferred information is a normal part of knowledge base development. Therefore, modifications can be safely done only by persons intimately familiar with the knowledge base. This factor not only limits the set of people who can make modifications, and of course precludes the success of an automatic knowledge acquisition system in which each rule is considered individually, but it also limits the size of the knowledge base, as even the best of knowledge engineers can retain familiarity with only a limited number of rules at a time. Again, CENTAUR removes this control knowledge from the inference rules, making it explicit in the prototypes and eliminating the necessity *for predicting complicated rule interactions*.

3.4 Control Structure

3.4.1 Problems with Rule-Based Systems

In both MYCIN and PUFF, questions are asked of the user when information is needed that can not be inferred by rules, so that choosing which rules to invoke in turn determines the questions that may be asked. Consultation questions are sometimes asked in an *unreasonable order* and *irrelevant questions* may even be asked because both systems follow a conservative strategy that explores *all* possible disease categories in a fixed order, so that no possibilities will be missed. Such a conservative strategy is necessary because there is no triggering mechanism to suggest the most likely disease, or to eliminate those that should not be

considered with the given data[4] Even if pulmonary function tests strongly indicate a certain disease, PUFF will begin exploring that disease only if it is first in order, and will first go through a series of questions to explore other diseases if it is not. Thus control of the system is not at all sensitive to the actual case data.

For example, in PUFF, there are four pulmonary disease categories that are explored in a fixed order. This order is set in PUFF's main "control" rule (shown in Figure 2.4) by the order of its premise clauses, and specifies that Obstructive Airways Disease will be the first disease to be explored, followed by Restrictive Lung Disease, *Diffusion Defect*, and finally the possibility that the patient has normal pulmonary function. This is illustrated on the left-hand side of Figure 3.8. Thus, even if the initial data[5] indicate that the patient is normal, the questioning still begins by exploring the possibility of OAD, then the other diseases, and entails asking many irrelevant questions.

Physicians using the systems were puzzled by these irrelevant questions, which seemed to indicate that the system's reasoning differed greatly from their own. Irrelevant questions were also irritating because of the extra time required to interact with the systems in order to perform an on-line consultation.

[4] It should be noted that it is possible to effect such a triggering mechanism in a rule-based system in which the premise clauses can be executed in an arbitrary order. In that case, a measure of belief can be associated with each premise clause. The clause with the highest measure of belief is then explored first. This kind of strategy is used in the PROSPECTOR consultation system for mineral exploration ([Duda, et al., 1978] and [Duda, et al., 1979]). However, in the EMYCIN systems being discussed here, where control knowledge is often represented implicitly in the order of the premise clauses, such a strategy would not be possible.

[5] The initial data discussed here are the "INITIALDATA parameters" described in Section 2.4.2, which are always determined first in the consultation.

CENTAUR's altered control structure is illustrated by the right-hand side of Figure 3.8. The triggering of prototypes by data values that serve as strong diagnostic clues to a physician allows immediate consideration of the indicated hypotheses, instead of routinely considering all possible hypotheses in turn. *CENTAUR focuses on the best prototype* as indicated by the data, and the disease first explored is, therefore, the one most strongly suggested by the initial pulmonary function test results. Further, diseases that are not suggested by the data will not be considered, unless they are suggested later in the consultation, saving many needless rule invocations and questions.

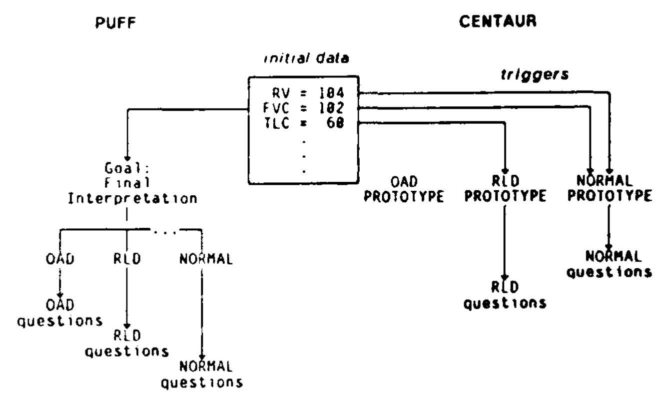

FIGURE 3.8 Control Structure Comparison--PUFF and CENTAUR

A second focusing strategy occurs once a prototype has been selected as the current best hypothesis, because only *rules associated with that prototype are used*

to *infer information* about a given clinical parameter. In PUFF and MYCIN there is a list of rules associated with each clinical parameter. Rules in this list are invoked when a parameter's value is needed. These parameters may be quite general, *applying to several diseases*. These general parameters also cause irrelevant questions in MYCIN and PUFF, since attempting to infer a value for a parameter causes questions to be asked about each disease in turn, some of which are not even suggested by the data.[6] For example, the parameter REVERSIBILITY in PUFF lists rules that infer information about the reversibility of the various pulmonary diseases. Therefore, even though only one disease is being considered, when a value for REVERSIBILITY is needed, REVERSIBILITY rules dealing with other diseases will be applied.[7]

By associating parameters with prototypes, (the prototype components), the PUFF rule lists are partitioned in CENTAUR into smaller lists of rules associated with components in the more narrowly defined context of the prototype. Further, the order in which values are obtained for the components, and thus the order in which questions are asked, is set explicitly for each prototype through an importance measure assigned to each component. Thus an expert can specify that the most

[6] "Screening clauses" sometimes are added to the premises of rules to avoid this problem. A screening clause tests for the condition causing the irrelevant question and causes the rule to fail when that condition is present. Screening clauses, however, introduce the problems of *implicit control in rules*, which have already been discussed.

[7] Alternatively, we could have defined five parameters dealing with reversibility, one for each disease. This would have sufficiently narrowed the context in which the rules were applied, but it would have greatly increased the amount of information dealing with reversibility of disease that needed to be represented. For example, rules referring to the concept of the reversibility of disease would now have to refer to all five parameters, and REVERSIBILITY rules that did apply to more than one disease would now have to be duplicated with separate *reversibility parameters*.

critical information be obtained first in exploring (or seeking to eliminate) a particular disease.

Irrelevant questions were also asked in PUFF because rules written to apply in more general situations contained clauses that did not apply in specific situations. In developing the PUFF knowledge base, there was a trade-off between writing a general rule that applied in a variety of situations and obtained enough information for all of the situations, and writing several smaller rules specific to each situation. In either case, the final result of the consultation would be the same, but the amount *of information obtained*, i.e., the number of questions asked, would differ. Because it was difficult to predict the precise situations in which the rule would be applied, the experts sometimes felt it was better to "play it safe" and write a more general rule that was certain to apply in all of the situations.

In CENTAUR, the precise situation or context in which the rule will be applied is determined by a prototype, with the result that the expertise in the rules can be very specific to each context. Associating rules explicitly with particular prototypes defines the different situations in which different versions of a particular rule will be applied, and has eliminated some of the irrelevant questions caused by applying excessively general rules. Further, in CENTAUR, a more general version of a rule can be associated with a more general prototype (such as the general PULMONARY-DISEASE prototype) to cover for any unexpected situations, while the more specific versions of the rule are associated with the more specific prototypes (such as the OBSTRUCTIVE AIRWAYS DISEASE prototype). In this way, a more specific rule is applied whenever possible, in the context of a more specific prototype.

3.4.2 Parallels to Physician's Reasoning

The process of medical problem solving has been discussed by many researchers (e.g., [Elstein, Shulman, and Sprafka, 1978] and [Kassirer and Gorry, 1978]) and it is widely felt that a sequence of suggesting hypotheses, acquiring further information, and then revising the hypotheses is, in fact, the problem-solving process used by most physicians. CENTAUR affords increased conceptual clarity then, in that the physician can understand what the system is doing. This leads to other advantages: the system's explanations of its performance during the consultation also seem more intelligible, for example.

3.5 Explaining System Performance

The ability to explain system performance was another design goal of the MYCIN project. Explanations are given in terms of which rules are currently being executed, which rules invoked those rules, and so on, in an unwinding of the chain of rules used to make conclusions about a clinical parameter. That is, *the quality of the explanations in rule-based systems is dependent on both the content of the rule and the system's control structure.* A good rule explanation requires the knowledge expressed in each rule to be suitable for an explanation of that rule, and the rule chaining to be understandable to the user. Unfortunately, as we have seen, much of the knowledge contained in a rule is not included explicitly in the rule premises, so that the explanations of rules are correspondingly insufficient. Clauses written to guide the invocation of other rules are not distinct from clauses containing purely descriptive medical expertise. Further, a control structure that results in the backward chaining of rules is sometimes difficult for a user to comprehend.

CENTAUR's explanation capability is improved not only because the overall control structure of the system is more focused, and in our experience easier to understand, but also because the content of the rules themselves is more clear to the user. Much of the knowledge that was represented implicitly in the rules has been represented explicitly elsewhere in the system.

CENTAUR's explanations are given in terms of the prototypes being considered as best hypotheses, as well as in terms of the rules being used to infer information. The prototype explanations define a context in which the rule is invoked and thus aid in *understanding* the *rule*. A full discussion of explanations in both PUFF and CENTAUR and examples of explanations in both systems is given in Chapter 7.

3.6 *Rule Knowledge Now Represented in Prototypes*

Some of the knowledge formerly represented in rules in PUFF is now represented in prototypes in CENTAUR. For example, rules that were written principally to guide invocations of other rules were replaced by control knowledge in prototype slots. This left a cleaner, more uniform rule base in which each inference rule represented some descriptive chunk of medical expertise. A user now examining the rule base is not confused by rules whose subject matter is medical, but which actually represent strategies for running the consultation.

Other knowledge that *had seemed difficult to represent in rules* was more easily represented in prototypes. In PUFF, for example, there are several parameters (TLC, MMF, FEV1/FVC, etc.) to be considered in determining the degree (MILD, MODERATE, MODERATELY-SEVERE, or SEVERE) of Obstructive Airways Disease

(OAD). Values are given for these parameters and an attempt is made to determine an overall conclusion of the degree of disease. In order to represent this process in *rules*, three sets of rules operating in three passes seemed necessary. The first set of rules, one for each parameter, concluded degrees of disease depending on the value of that parameter. The second set combined these conclusions, weighting some parameters more than others. In the *third set, an attempt was made to resolve* any obvious conflicting conclusions, such as a final conclusion of both MILD and SEVERE OAD in the same patient. In order to ensure that the rules would be invoked in three separate passes, extra clauses were added to the rule premises to control their invocation.

The final set of rules seemed confusing, even to our expert, and modifying one or more of them often required identifying necessary changes in all three sets. English translations of the rules did not clearly indicate the strategies that were being represented, and the system's *explanation of it's reasoning*, given in terms of the rules it had used, was correspondingly weak. Consider, for example, the following PUFF rule:

> If the degree for OAD is NONE, and the degree for OAD by the MMF is greater than or equal to MILD, then the degree for the OAD is MILD.

The medical expertise expressed in this rule is not apparent. In order to understand this rule, it is necessary to see it as only one part of the group of rules that together *determine the degree of OAD in a patient*. The first clause of the rule, "If the degree for OAD is NONE" requires that the degree for OAD already be

determined, which first invokes other rules. This rule, then, is invoked with the second set of rules as discussed above. To satisfy the first premise clause, the *degree parameters, taken together*, must have indicated a degree of NONE for OAD. The second clause tests one of the degree parameters, the MMF measurement, and essentially gives more weight to that parameter by overruling the initial conclusion of degree *NONE* in *favor of a degree of MILD*.

With the addition of prototypes to the knowledge base, all of the rules in these three sets were eliminated. Separate prototypes were created for each possible degree of OAD (MILD-OAD, MODERATE-OAD, MODERATELY-SEVERE-OAD, and SEVERE-OAD). Prototype components were created for each of the parameters used in *determining the degree of disease*, and the Importance Measures were set to indicate the weight with which to consider each parameter in determining the degree of disease. Each degree prototype thus represents a pattern of parameters and values, and the system attempts to *determine the best match of actual case data* to one of these patterns.

Altering the weight with which one of the parameters is considered in CENTAUR merely requires changing one number. Considering a new parameter entails adding a component. In short, modifications are less complicated to make, and the knowledge is more clearly represented. As this example illustrates, typical data patterns were easily represented in prototypes.

Chapter 4

CENTAUR Example

4.1 Introduction

The CENTAUR consultation presented in this chapter was *run on the same* pulmonary function test data used for the PUFF example in Chapter 2. As before, annotations are enclosed in parentheses and written in *italics*. User responses are in **BOLDFACE CAPITALS** and follow double asterisks (**). *Some of the most important* features of the program are indicated by annotations written in ***boldface italics*** at the right margin. Many of these features are discussed in later chapters.

A portion of the prototype network, useful for *understanding this example*, is given in Figure 4.1 for reference. At any one time during the consultation, there is a single **Current Prototype** which is the focus of attention for system processing. The CONSULTATION prototype at the top of the prototype network is the initial Current Prototype for performing consultations. It controls the stages of the consultation and sets certain user options, such as the strategy for selecting which prototype to pursue from the hypothesis list of prototypes. The next prototype chosen, the PULMONARY-DISEASE prototype, controls the acquisition of an initial set of data. The consultation then continues as disease, degree, and subtype prototypes in the network are selected as possible matches to the data. In this example, Obstructive Airways Disease (OAD), Severe Obstructive Airways Disease, and Asthma are confirmed as matching the data.

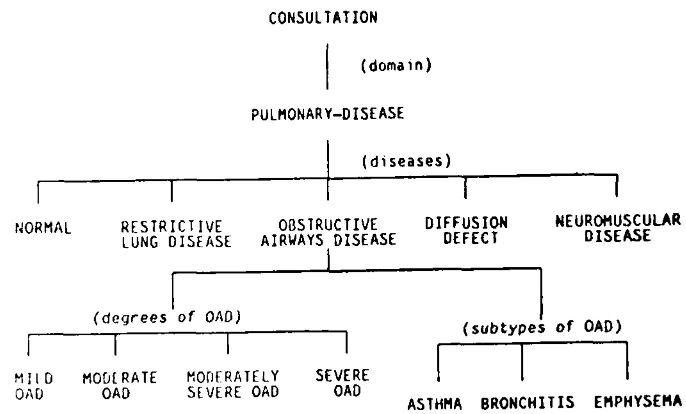

FIGURE 4.1 A Portion of the Prototype Network

4.2 The Example

```
----------------------------------------------------------------------
18-Nov-79 23:22:24
[consultation of 23-Feb-77 11:55AM]

*CENTAUR*
```

*(The CONSULTATION prototype is the first prototype selected. The tracing level, agenda printing option, and consultation strategy questions are asked to "fill in" user options for the consultation. A medium level of tracing is chosen to indicate some of the internal features of the program. Tracing information is printed by the program in [**brackets**].)*

```
Tracing Level (0-3):
** 2
Agenda Printing?
** No
Consultation Strategy:
** ?
```

(User specifies strategy for selecting among possible hypotheses. (See Section 6.7.1.1.))

```
Please select from among CONFIRMATION, ELIMINATION, FIXED-ORDER
Consultation Strategy:
** CONFIRMATION
```

(After the CONSULTATION prototype is filled in, the PULMONARY DISEASE prototype is chosen as the next Current Prototype. To fill in this prototype, an initial set of pulmonary function test results is requested from the user in questions 1 through 10. As the initial data are entered, prototypes are suggested as being likely matches to the given data values. For example, in question 6, the OAD prototype is suggested by the value 40 for the FEV1/FVC ratio. The Certainty Measure (CM) indicates on a numerical scale (from -1000 to 1000) the degree of certainty with which the prototype is indicated by the data.)

```
--------PATIENT-7--------
1) Patient's identifying number:
** 7446
2) referral diagnosis:
** ASTHMA
  [Trigger for ASTHMA and CM 900]
3) RV/RV-predicted:
** 261
4) TLC(body box)observed/predicted:
** 139
5) FVC/FVC-predicted:
** 81
  [Trigger for NORMAL and CM 500]
6) FEV1/FVC ratio:
** 40
  [Trigger for OAD and CM 900]
```


(Prototypes are triggered by the initial data values.)

```
7) the DLCO/DLCO-predicted:
** 117
  [Trigger for NORMAL and CM 700]
8) Change in FEV1 (after dilation):
** 31
9) MMF/MMF-predicted:
** 12
  [Trigger for OAD and CM 900]
10) The slope F5025:
** 9
  [Trigger for OAD and CM 900]
```

(A summary of the suggested prototypes is printed. In this case, three different prototypes were triggered by the initial data. NORMAL and OAD are disease prototypes and will be considered next. ASTHMA is a subtype of OAD and will be considered later, when subtypes of OAD are being explored.)

```
Suggested Hypotheses
   Hypothesis: ASTHMA, CM: 900, Reason: RDX was ASTHMA
   Hypothesis: NORMAL, CM: 500, Reason: FVC was 81
   Hypothesis: OAD, CM: 900, Reason: FEV1/FVC was 40
   Hypothesis: NORMAL, CM: 700, Reason: DLCO was 117
   Hypothesis: OAD, CM: 900, Reason: MMF was 12
   Hypothesis: OAD, CM: 900, Reason: F5025 was 9
```

(Although there are five possible, more specific disease prototypes for the PULMONARY-DISEASE prototype, only the two that were triggered by the initial data are selected as possibilities to pursue. These prototypes are filled in with the data values that are already known in the case.)

(Only triggered hypotheses are chosen to pursue.)

```
More Specific hypotheses chosen: NORMAL, OAD
   [New prototypes being filled in ... NORMAL, OAD]
```

(Any data values that are not consistent with the values expected for that disease prototype are noted as Surprise Values, and the CM for that prototype is lowered. In this case, five of the data values are not consistent with the NORMAL pulmonary function prototype. Similarly, the CM of each prototype is raised for values that are consistent with the expected values set in the prototype, although this is not shown in the

print-out. The function used for combining Certainty Measures is discussed in Section 5.2.5.)

(Values not consistent with a prototype are marked.)

```
'Surprise Value'  261 for RV in NORMAL, CM: 700
'Surprise Value'  139 for TLC in NORMAL, CM: 400
'Surprise Value'  40 for FEV1/FVC in NORMAL, CM: -166
'Surprise Value'  12 for MMF in NORMAL, CM: -499
'Surprise Value'  9 for F5025 in NORMAL, CM: -699
```

(The Hypothesis List of triggered prototypes is then ordered according to the CM of the prototypes. The Confirmation Strategy selected for this consultation attempts to confirm the most likely prototype, in this case the OAD prototype.)

(The consultation strategy dictates which prototype is selected from the hypothesis list.)

```
Hypothesis List:  (OAD 999) (NORMAL -699)

I am testing the hypothesis that there is Obstructive Airways Disease.
```

(In order to instantiate the OAD prototype, several more components must have values. If there are rules associated with these components that can be used to deduce their values, they will be tried. Otherwise, the user is asked for the value.)

(The context (OAD) of the following questions is stated explicitly.)

```
Components of OAD chosen to trace:  F25 D-RV/TLC FEV1 D-FEV1/FVC
                                    COUGH SPUTUM
```

(The user may ask for expected responses to a question by typing a "?". Note the more helpful information provided by the OAD context.)


```
11) The flow F25:
** ?
What is the F25 of 7446?
Expected responses are: number
Furthermore, for Obstructive Airways Disease it is expected
   that the value is less than 60.
Enter HELP for list of user options.

** 45
12) RV/TLC Difference:
** 25
13) FEV1/FEV1-predicted:
** 79
14) FEV1/FVC Difference:
** UNKNOWN
```

(The user may also enter "UNKNOWN" in response to a question.)

```
15) The severity of coughing:
** NONE
16) The degree of sputum production:
** NONE
```

(Again, surprise values are noted for all prototypes currently on the hypothesis list. A judgement is then made about whether the current prototype, OAD, is a close enough match to the data values to be confirmed in this case.)

```
'Surprise Value!  45 for F25 in NORMAL, CM: -819
```

(Intermediate conclusions are stated explicitly. The matching criteria are discussed in Section 5.2.2.)

```
Obstructive Airways Disease meets the matching criteria.
Based on the data provided, it is confirmed that there is
   Obstructive Airways Disease.
```

(Control knowledge associated with the OAD prototype specifies that the Degree of OAD should be determined next, followed by the Subtype of OAD. No degree prototypes were triggered by the data values, so all of them are selected as possible hypotheses to be filled in with the data values in the case, and the consultation continues.)

```
More Specific hypotheses chosen: MILD-OAD, MODERATE-OAD,
   MODERATELY-SEVERE-OAD, SEVERE-OAD

    [New prototypes being filled in ... MILD-OAD, MODERATE-OAD,
    MODERATELY-SEVERE-OAD, SEVERE-OAD]

'Surprise Value' 261 for RV in MILD-OAD, CM: -400
'Surprise Value' 12 for MMF in MILD-OAD, CM: -464
'Surprise Value' 9 for F5025 in MILD-OAD, CM: -678
'Surprise Value' 45 for F25 in MILD-OAD, CM: -886
'Surprise Value' 25 for D-RV/TLC in MILD-OAD, CM: -883
'Surprise Value' 261 for RV in MODERATE-OAD, CM: -400
'Surprise Value' 12 for MMF in MODERATE-OAD, CM: -464
'Surprise Value' 9 for F5025 in MODERATE-OAD, CM: -678
'Surprise Value' 25 for D-RV/TLC in MODERATE-OAD, CM: -677
'Surprise Value' 79 for FEV1 in MODERATE-OAD, CM: -886
'Surprise Value' 261 for RV in MODERATELY-SEVERE-OAD, CM: -400
'Surprise Value' 12 for MMF in MODERATELY-SEVERE-OAD, CM: -464
'Surprise Value' 9 for F5025 in MODERATELY-SEVERE-OAD, CM: -678
'Surprise Value' 45 for F25 in MODERATELY-SEVERE-OAD, CM: -886
'Surprise Value' 25 for D-RV/TLC in MODERATELY-SEVERE-OAD, CM: -883
'Surprise Value' 79 for FEV1 in MODERATELY-SEVERE-OAD, CM: -929
'Surprise Value' 45 for F25 in SEVERE-OAD, CM: 798
'Surprise Value' 79 for FEV1 in SEVERE-OAD, CM: 756

Hypothesis List: (SEVERE-OAD 756) (MILD-OAD -805)
   (MODERATE-OAD -806) (MODERATELY-SEVERE-OAD -929)
```

(Again, the Confirmation Strategy dictates that the most likely prototype be selected from the hypothesis list as the next Current Prototype.)

I am testing the hypothesis that there is Severe Obstructive Airways Disease.

(In evaluating the match of Severe OAD to the data, facts that are inconsistent with the expected data values in the prototype are noted for the user. However, the match of the prototype to the data is still good enough to confirm it in this case.)

(Inconsistent facts are noted.)

```
Facts inconsistent with Severe Obstructive Airways Disease are:
   The fev1/fev1-predicted ratio of PATIENT-7: 79
   The f25 of PATIENT-7: 45

Based on the data provided, it is confirmed that there is
   Severe Obstructive Airways Disease.
```


(Facts that are known in the case, and that represent expected values in either SEVERE-OAD or OAD, are marked "accounted for" by those prototypes. CENTAUR's Fact-Residual Rules represent a set of expertise that uses these remaining, unaccounted for facts to help guide further processing. These rules are applied after prototypes have accounted for facts. Any residual facts, and a list of prototypes that could account for them, are then printed for the user.)

```
[Facts marked Accounted For by SEVERE-OAD.]

[Facts marked Accounted For by OAD.]

[Fact-Residual Rules being applied ...]
```

(In this case, there is only one residual fact, the DLCO measurement. CENTAUR notes that the value 117 is an expected value in three prototypes. After inspecting the system agenda, CENTAUR defers attempting to account for this residual fact because there is still another set of prototypes, the OAD subtype prototypes, to be explored. The ASTHMA prototype is one of these subtype prototypes, and it also could account for this fact.)

(Residual facts are used to guide system processing.)

```
Facts remaining to be accounted for and possible hypotheses to
   account for them are:
   Fact: DLCO was 117   Hypotheses: ASTHMA NORMAL SUPER-NORMAL
```

(The user is notified that CENTAUR was able to determine a degree for OAD. If a task fails, the user is given the option of terminating the consultation. Failure to complete a task can indicate a data error or a gap in the knowledge base that would invalidate the system's conclusions. If the user desires to continue the consultation even though a task has failed, then rules associated with a more general prototype are used to conclude the task. For example, if the task to determine a degree prototype for OAD failed, then rules associated with the OAD prototype itself would be used to conclude a degree for OAD.)

(Task status is printed for the user.)

```
The task to determine the DEGREE of OAD succeeded.
The solutions were:   SEVERE-OAD
```

(The task to determine a subtype for OAD is executed next. ASTHMA was suggested by the initial data, so it is selected as the most likely subtype prototype to consider.)

```
More Specific hypotheses chosen: ASTHMA

     [New prototypes being filled in ... ASTHMA]

Hypothesis List:  (ASTHMA 930)

I am testing the hypothesis that there is Asthma.
```

(Again, there are components of the ASTHMA prototype that need values before ASTHMA can be confirmed in this patient. In this case, there are rules used to infer a value for DEG-REV. Trying those rules requires additional information, which causes questions 17 and 18 to be asked.)

```
Components of ASTHMA chosen to trace:  DEG-REV SPUTUMPURULENCE

17) Change in MMF (after dilation):
** 100
18) The change in resistance (after dilation):
** 2.0
19) Sputum purulence:
** ?
Is there sputum purulence?
Expected responses are: YES or NO
Furthermore, for Asthma it is expected that there is not
   sputum purulence.
Enter HELP for list of user options.

** NO

Asthma meets the matching criteria.

Based on the data provided, it is confirmed that there is Asthma.

     [Facts marked Accounted For by ASTHMA.]

     [Fact-Residual Rules being applied ...]
```

(At this point, there are no more residual facts, and no more sets of prototypes to be explored, so the system prints its interim conclusions.)


```
INTERIM RESULTS

Confirmed Hypotheses: Asthma, Severe Obstructive Airways Disease,
     Obstructive Airways Disease
```

(This CONSULTATION prototype specifies that a Refinement Stage be included which applies a further set of domain expertise (contained in the Refinement Rules) to make further conclusions and recommendations about the program's interim results. These rules are associated with the confirmed prototypes, and may result in more questions being asked of the user. Questions 20, 21, and 22 are asked because of information needed for various refinement rules.)

(The consultation moves into the refinement stage.)

```
     [Refinement Rules being applied ...]

20) The number of pack-years of smoking:
** 17
21) The number of years ago that the patient stopped smoking:
** 0
22) The degree of dyspnea:
** NONE
```

(Summary Rules associated with the confirmed prototypes are executed next. These rules merely summarize information in the prototypes and do not require any new information to be obtained from the user.)

```
     [Summary Rules being applied ...]
```

(The Action Slots of the confirmed prototypes control the printing of findings associated with the prototype. Notice that the findings about each disease prototype are now grouped together. SEVERE-OAD and ASTHMA do not have ACTION Slots.)

```
[Action slot of OAD being executed ...]

Conclusions: the findings about the diagnosis of obstructive
    airways disease are as follows:
      Elevated lung volumes indicate overinflation.
      The RV/TLC ratio is increased, suggesting a severe degree
         of air trapping.
      Forced Vital Capacity is normal but the FEV1/FVC ratio is
         reduced, suggesting airway obstruction of a severe degree.
      Low mid-expiratory flow is consistent with severe airway
         obstruction.
      Obstruction is indicated by curvature of the flow-volume
         loop which is of a severe degree.
      Following bronchodilation, expired flow shows excellent
         improvement as indicated by the change in the FEV1.
      Following bronchodilation, expired flow shows excellent
         improvement as indicated by the change in the MMF.
      Reversibility of airway obstruction is confirmed by
         improvement in airway resistance following
         bronchodilation.
```

(The Action Slot of the PULMONARY-DISEASE prototype controls the printing of the prototype summary, the final conclusions about the pulmonary function tests, and the diagnosis of pulmonary disease in the patient. Notice that the findings that suggested the prototype are listed, as well as those that are consistent and inconsistent with the prototype, and those that were marked accounted for by the prototype. Indentation indicates the level of the prototype in the prototype network.)

```
[Action slot of PULMONARY-DISEASE being executed ...]

   -----Prototype Summary-----

  -----Obstructive Airways Disease-----

  Obstructive Airways Disease was suggested by the following
     findings
     The fev1/fvc ratio of PATIENT-7:  48
     The mmf/mmf-predicted ratio of PATIENT-7:  12
     The f5025 of PATIENT-7:  9

  In addition, Obstructive Airways Disease is consistent with
     The tlc/tlc-predicted ratio of PATIENT-7:  139
     The rv/rv-predicted ratio of PATIENT-7:  261
     The f25 of PATIENT-7:  45
     The observed-predicted difference in rv/tlc of PATIENT-7:  25
     The fev1/fev1-predicted ratio of PATIENT-7:  79
     The severity of coughing of PATIENT-7:  NONE
     The degree of sputum production of PATIENT-7:  NONE
```



```
The Obstructive Airways Disease accounts for the following
   findings:
   The referral diagnosis of PATIENT-7
   The fev1/fvc ratio of PATIENT-7
   The f25 of PATIENT-7
   The fev1/fev1-predicted ratio of PATIENT-7
   The severity of coughing of PATIENT-7
   The degree of sputum production of PATIENT-7

   -----Severe Obstructive Airways Disease-----

   Severe Obstructive Airways Disease is consistent with
      The rv/rv-predicted ratio of PATIENT-7:  261
      The tlc/tlc-predicted ratio of PATIENT-7:  139
      The observed-predicted difference in rv/tlc of
         PATIENT-7:  25
      The fvc/fvc-predicted ratio of PATIENT-7:  81
      The mmf/mmf-predicted ratio of PATIENT-7:  12
      The f5025 of PATIENT-7:  9

   Findings inconsistent with Severe Obstructive Airways
      Disease are
      The fev1/fev1-predicted ratio of PATIENT-7:  79
      The f25 of PATIENT-7:  45

   The Severe Obstructive Airways Disease accounts for the
      following findings:
      The rv/rv-predicted ratio of PATIENT-7
      The tlc/tlc-predicted ratio of PATIENT-7
      The fvc/fvc-predicted ratio of PATIENT-7
      The mmf/mmf-predicted ratio of PATIENT-7
      The f5025 of PATIENT-7
      The observed-predicted difference in rv/tlc of PATIENT-7
      The tlc(body box) of PATIENT-7

   -----Asthma-----

   Asthma was suggested by the following findings
      The referral diagnosis of PATIENT-7:  ASTHMA

   In addition, Asthma is consistent with
      The degree of reversibility of airway obstruction
         of PATIENT-7:  EXCELLENT
      The dlco/dlco-predicted of PATIENT-7:  117
      Whether there is sputum purulence:  YES

   The Asthma accounts for the following findings:
      The dlco/dlco-predicted of PATIENT-7
      The degree of reversibility of airway obstruction
         of PATIENT-7
      Whether there is sputum purulence
```

(If facts had remained unaccounted for at the end of the consultation, they would have been indicated to the user at this point. CENTAUR can indicate facts that are not accounted for because the representation of prototypical cases specifies expected findings.)

```
All facts have been accounted for by the confirmed prototypes.

Conclusions:
        Smoking probably exacerbates the severity of the patient's
            airway obstruction.
        Discontinuation of smoking should help relieve the symptoms.
        Good response to bronchodilators is consistent with an asthmatic
            condition, and their continued use is indicated.
        The high diffusing capacity is consistent with asthma.

Pulmonary Function Diagnosis:
        Severe Obstructive Airways Disease.
        Asthmatic type.

Consultation Finished.
```

4.3 What the Example Shows

In Section 2.6, we discussed the PUFF example and pointed out that some desirable information, such as the context of each question and the stages in a consultation, can not be printed because it is not explicitly represented in the PUFF rules. Intermediate conclusions are not shown in PUFF because there is no representation of major reasoning steps, as distinct from inference rules. Only a detailed tracing of *all* rule conclusions is available. In CENTAUR, the current hypothesis is stated, for example, *I am testing the hypothesis that there is ASTHMA*, to give the user a context for the questions being asked. Intermediate conclusions, such as *Based on the data provided, it is confirmed that there is Asthma* are also displayed to inform the user of the program's progress in diagnosing a disease. The

various stages of the consultation, such as the stage in which Refinement Rules are applied to an interim diagnosis, are announced. Thus the user is aware, for example, that the program's questions are attempting to determine refinement knowledge about an already confirmed disease, not *exploring* other diseases.

PUFF also lacked the ability to deal with inconsistent or erroneous information because it had no representation of prototypical disease patterns. CENTAUR, on the other hand, prints inconsistent data both during the consultation and in the prototype summary to inform the user of possible problems in the interpretation. CENTAUR also uses seemingly inconsistent data to guide further reasoning during the consultation.

PUFF's final interpretation did not relate the facts in the case to the *interpretation statements*, but the format of CENTAUR's prototype summary makes this connection explicit. CENTAUR's final interpretation not only prints the facts that are accounted for by each confirmed prototype, but also displays the interpretation statements in groups associated with each prototype.

A comparison of the number of questions asked by CENTAUR and PUFF on twenty cases is given in *Chapter 8*. In the sample case presented in this chapter, CENTAUR explored fewer diseases than PUFF. There were, however, more questions asked in the CENTAUR consultation because there are more concepts represented about OAD (in the prototype components). For example, the component SPUTUM PURULENCE in the ASTHMA prototype is not represented in PUFF. Considering only the subset of rules that are represented in both systems, CENTAUR executed fewer rules because only rules associated with OAD and its degrees and subtypes were tried. In this case, all of the data were consistent with this diagnosis, so no other

prototypes were tried. If any facts had remained to be accounted for after the OAD diagnosis was made, other prototypes then would have been explored in an attempt to determine whether there were multiple diseases. In PUFF, rules associated with all four of the represented diseases and all three of the possible OAD subtypes were tried. Thus the prototypes allowed CENTAUR to focus on the most relevant rules in each case.

Chapter 5

CENTAUR Knowledge Representation

5.1 Introduction

Knowledge is represented in CENTAUR using both production rules and frames. The frame-like structures in CENTAUR are **prototypes, prototype components,** and **facts**. Following frame terminology [Minsky, 1975], each of these structures contains slots of information associated with it. (Templates for all three data structures, including the complete list of slots, are presented in Appendix C for reference.) One of the slots in the prototype frame is the **Components** slot. Because each component is itself a frame, the value of the Components slot in the prototype is actually a set of "sub-frames" of knowledge. (Recall that the relationship between prototypes and components was diagrammed in Figure 1.2.) *Object-level* domain knowledge is specified in these prototype components, each of which represents one of the principal characterizing features of the prototype.

Other slots in the prototype represent *meta-level* knowledge, or knowledge about the prototype itself. These include slots that specify control knowledge, slots that specify sets of production rules to be used during the consultation, and slots that give general information about the prototype, such as a set of pointers relating the prototype with other prototypes in the knowledge base.

Some of the possible prototype and component slots are shown in Figures 5.1 and 5.2 respectively. An instantiation of some of the prototype and component slots

for the OBSTRUCTIVE AIRWAYS DISEASE (OAD) prototype is presented in Figure 5.3. The LISP representation and English translation of the entire OAD prototype are given in Appendix D.

This chapter describes the possible prototype slots with emphasis on the components and control slots. The five different groups of production rules are presented, with examples of each. The representation of dynamic data as **Facts** is discussed, and a sample of a *single fact* at *various* stages of the consultation is shown. Finally, a comparison is made between CENTAUR's knowledge representation and that of two other frame-oriented medical diagnosis systems, PIP ([Pauker et al., 1976], [Szolovits and Pauker, 1976], [Pauker and Szolovits, 1977], and [Szolovits and Pauker, 1978]) and INTERNIST ([Pople, 1975] and [Pople, 1977]).

Slot Name	Type of Information	
Author Date Source	Bookkeeping *Information*	General Slots
More General More Specific Alternate	Pointers to Other Prototypes	General Slots
Explanation Hypothesis	English Phrases	General Slots
To-Fill-In If-Confirmed If-Disconfirmed Action		Control Slots
Fact-Residual Rules Refinement Rules Summary Rules		Rule Slots
Components		

FIGURE 5.1 Possible Prototype Slots

Plausible Values
Default Value
Possible Error Values
Inference Rules
Importance Measure

FIGURE 5.2 Component Slots
(some or all may be filled in)

Obstructive Airways Disease Prototype

Author: Aikins
Date: 27-OCT-78 17:13:29
Source: Fallat
Pointers: (degree MILD-OAD) (degree MODERATE-OAD) ...
(subtype ASTHMA) (subtype EMPHYSEMA) ...
Hypothesis: There is Obstructive Airways Disease.

If-Confirmed: Deduce the degree of OAD
Deduce the subtype of OAD
Action: Deduce any findings associated with OAD
Print the findings associated with OAD

Fact-Residual Rules: RULE157, RULE158, ...
Refinement Rules: RULE036, RULE038, RULE039, ...
Summary Rules: RULE053, RULE054, RULE055, RULE083, ...

Components:

Total Lung Capacity	**Plausible Values:** >100 **Importance Measure:** 4
Reversibility	**Inference Rules:** RULE019, RULE020, RULE022, RULE025 **Importance Measure:** 0

.
.
.

FIGURE 5.3 Sample Slot Values for OAD Prototype and Two OAD Components (from Appendix D)

5.2 Prototypes

The CENTAUR knowledge base includes 24 *prototypes*, 21 of which represent disease patterns in the pulmonary function domain. The remaining prototypes represent the Consultation task itself (**CONSULTATION** prototype), a Prototype Knowledge Review task (**REVIEW** prototype), and knowledge common to any pulmonary function interpretation (**PULMONARY-FUNCTION** prototype). All of the prototypes are linked together in a network where the links specify the relationships between prototypes. A portion of the prototype network was shown in Figure 4.1.

5.2.1 Component Slots

Each prototype contains six to eight prototype components. In the pulmonary function task, where prototypes represent the pulmonary diseases, each component represents some characteristic feature of *a pulmonary disease*. For example, in the OAD prototype, there are component slots for the pulmonary function tests useful in characterizing a patient with OAD; two of these are shown in Figure 5.3. For example, since the TOTAL LUNG CAPACITY of a patient with OAD is typically higher than that of a person with normal pulmonary function, there is a component, TOTAL LUNG CAPACITY, with a range of **PLAUSIBLE VALUES** that are characteristic of a person with OAD. The syntax for the plausible values slot is shown in Figure 5.4.

Syntax:
```
(((Condition 1) (Action 1))
 ((Condition 2) (Action 2))
 ((Condition 3) (Action 3)) ...)
```

Sample: OAD prototype, TLC component

Plausible Values:
```
If: The value is greater than or equal to 100
Then: No action is indicated
```

LISP: (((GREATEQ* $VALUE 100)))

FIGURE 5.4 Plausible Values Syntax and Sample for a Total Lung Capacity Component

Each plausible value is a set of condition-action pairs.[1] Some of the possible conditions and actions are as follows:

Conditions	Actions
If the given value is:	Then:
▮ Equal or Not Equal to a value	▮ Make a conclusion
▮ Greater than a value	▮ Print a statement
▮ Less than a value	▮ Trigger a prototype
▮ Within a given range of values	▮ Do Nothing
▮ One of a set of values	
▮ Any value	

Generally, in a pulmonary function problem, most plausible value slots include a single condition, which is a predicate specifying a range of values, with no action indicated. It has been useful, however, to have the flexibility to perform some action

[1] The plausible values are rules in effect, as are the possible error values.

when the given condition is satisfied, and all of the possible actions listed above are used in the prototypes.

In addition to a set of plausible values, that is, values consistent with the hypothesis represented by the prototype, the component frames may have other slots as well. (The ways in which this information is used is discussed in Chapter 6.) There may be one or more **POSSIBLE ERROR VALUES**, that is, values that are inconsistent with the prototype or that might have been specified by the expert to check what he considers to be a measurement error. The syntax for possible error values is the same as that for plausible values. Generally, the action associated with a possible error value either triggers another prototype (if the value indicates that another prototype may be a better match to the data) or prints a statement to inform the user of the error value and to suggest a possible fix. For example, the statement might specify that one of the pulmonary function tests *be repeated* to ensure accuracy.

Data values are classified as Plausible Values, Possible Error Values or **SURPRISE VALUES** using the ranges of values specified in these slots. Surprise Values are all of those values that are neither Plausible Values nor Possible Error Values. They indicate facts that cannot be accounted for by the hypothesis represented by a given prototype.

A component may also have a **DEFAULT VALUE** independent of the other values. All of the components in a disease prototype, with their default values, form a picture of the typical patient with the disease. These default values are useful when the system is explaining the prototypical knowledge. They are also used when a value is

needed for a component that can not be inferred by *rules* and *is not known* to the user.

There is also a slot whose value is a list of **INFERENCE RULES** used to infer a value for the component.[2] For example, there are four rules associated with the REVERSIBILITY component in Figure 5.3. These rules are tried when a value is needed for the REVERSIBILITY of OAD. These are CENTAUR's **Object-level Inference Rules**. One distinction between the organization of rule knowledge in PUFF and CENTAUR is apparent here. In PUFF, the Inference Rules are grouped according to parameters in the rule conclusions. When a value is needed for a parameter, the corresponding list of rules is retrieved, and the rules on that list are used in an attempt to infer the value. In CENTAUR, these lists of rules are associated with the component *in the more narrowly defined context of the prototype*. This not only makes explicit the context in which each rule is applied, but it also results in a smaller set of applicable rules when a value is needed for a component. For example, the PUFF system has a list of rules associated with the clinical parameter REVERSIBILITY (which are tried in order to determine if the effects of the disease are reversible), but in *CENTAUR*, there is a *REVERSIBILITY* c nponent for each disease prototype where reversibility of disease is a consideration. Only that subset of rules which are useful for obtaining a value for REVERSIBILITY *when a particular disease is being considered* are listed with the REVERSIBILITY component in that disease prototype.

Finally, each component has an **IMPORTANCE MEASURE** (from 1 to 5) that

[2] These are analogous to the "To-Fill" or "If-Needed" procedures used in many frame systems, e.g., GUS [Bobrow, et al., 1977] and NUDGE [Goldstein and Roberts, 1977].

indicates the relative importance of the component in characterizing the situation described by that prototype. The importance measures are used (a) to determine if the prototype matches the data, and (b) to select a component to be instantiated in the absence of prototype-specific control information, as described in the next section. An importance measure of 0 is used to indicate components whose values are not considered in the matching process. For example, the REVERSIBILITY component in Figure 5.3 has an importance measure of 0 because a patient may have OAD irrespective of whether the disease process is reversible.

5.2.2 Prototype Control Slots

Four of the slots associated with each prototype contain a set of one or more [illegible] clauses that are executed by the system at specific times to control the consultation. These clauses express actions to be taken by the system in order to instantiate a prototype (TO-FILL-IN slot), upon confirmation of a prototype (IF-CONFIRMED slot), in the event that a prototype is disconfirmed (IF-DISCONFIRMED slot), [illegible] have been derived (ACTION slot). The current set [illegible] clauses [illegible] clauses that specify sets of components for [illegible] If-Confirmed and If-Disconfirmed clauses that [illegible] to be explored next, and Action clauses that print [illegible]. Each control slot is essentially the action part of a rule [illegible] current situation matches the situation described by this [illegible] invoked at a point during the consultation, as defined [illegible] (see Section 6.4).

When a prototype is first selected as the Current Prototype, the system executes the clauses in the To-Fill-In slot of that prototype. The information in this slot indicates what components should have values, and in what order they should be determined. If no To-Fill-In slot is associated with a prototype (as in the OAD prototype in Figure 5.3), the default procedure is to instantiate components in the order of their importance measures, choosing those with the highest importance measures first. This default procedure is also specified in the Consultation prototype, and can be changed if desired.

Once the prototype has been instantiated,[3] the system decides whether the prototype should be confirmed as matching the given data. Confirmation is by comparison of a *Match Measure (MM)* with a numerical threshold which is set in the Consultation prototype.[4] The algorithm to calculate the *Match Measure* is defined as follows:

[illegible] that has a value, if the given component value is a [illegible]

[illegible] (MM)

[illegible] Measure of that component and [illegible] combining function described in [illegible]

[illegible]

[illegible] (MM)

[3] [illegible]

[4] [illegible] criterion with each individual prototype [illegible] necessary for the pulmonary diagnosis problem [illegible] used in the formula is not critical; empirically it gives the performance that was desired.

Information in the If-Confirmed or If-Disconfirmed slots provides prototype-specific instructions on how to proceed with a consultation. Similarly, the Action slot specifies steps to be taken for each confirmed prototype when final results are being presented. A discussion of the advantages of this scheme is given in Section 6.6.

5.2.3 Prototype Rule Slots

There are five different groups of rules used in the CENTAUR system. The Inference Rules, associated with individual components, were discussed in Section 5.2.1. TRIGGERING RULES are antecedent rules associated with individual parameters, and will be discussed in Section 5.3. The three other types of rules are values of prototype rule slots. They are used at specific stages in the consultation in an attempt to account for residual facts (**FACT-RESIDUAL RULES**), to suggest further lab tests or otherwise refine the diagnosis (**REFINEMENT RULES**), and to summarize the information in the prototype (**SUMMARY RULES**). Each prototype has three rule slots, corresponding to these three rule groups, which list the rules in each group for that prototype. Besides premise and action clauses, each of these rules has a property, PROTOTYPE, specifying the prototype with which it is associated. The PROTOTYPE property indicates explicitly the context in which the rule is applied, and is useful information when the rules are examined independently of the prototypes. Section 5.3 includes samples from all rule groups; a discussion of how rules are acquired in CENTAUR is given in Section 7.6.4.

5.2.4 General Information Slots

In addition to the domain-specific components and prototype control slots, each prototype also contains slots for general information associated with it. These include bookkeeping information (**AUTHOR, DATE,** and **SOURCE**), English phrases used in communicating with the user (**EXPLANATION** and **HYPOTHESIS**), and pointers to other prototypes in the prototype network. The pointers have the form (**LINK PROTOTYPE**), where the link specifies the relationship between the two prototypes. For example, in the OAD prototype in Figure 5.3, the pointers include the entry *(Subtype Asthma)*, to indicate that Asthma is a subtype of OAD. In CENTAUR, there are pointer slots for **MORE SPECIFIC, MORE GENERAL,** and **ALTERNATE** prototypes. This information is used during the consultation by control tasks that indicate categories of more specific, more general, or alternate prototypes to explore. For example, the task *determine the subtype of OAD* would retrieve the set of subtype prototypes from the OAD More Specific Slot. All of the information in these slots is static information, known before the consultation is run.

5.2.5 Certainty Measures

There are also slots containing dynamic information that are filled in with values as the consultation proceeds. Figure 5.5 summarizes the dynamic information slots and gives an example of values for these slots in the Obstructive Airways Disease prototype. Each prototype has a **CERTAINTY MEASURE** (from -1000 to 1000) that indicates how certain the system is that the prototype matches the data in each case. The Certainty Measure is used in selecting the prototype that

represents the current best hypothesis. Certainty Measures are similar to the Certainty Factors associated with rules and discussed in Section 2.4.3, and the algorithm for combining two Certainty Measures is the same as the one presented for Certainty Factors. However, these are two independent measures, with Certainty Factors applying to rule conclusions (the certainty of an individual fact), and Certainty Measures to the hypothesis represented by a prototype.

Certainty Measures (CMs) are initially set to 0, indicating that no prototype is more likely than any other initially. In pulmonary function diagnosis, Certainty Measures are initially set by Triggering Rules associated with the *Referral Diagnosis* parameter. For example, if the referral diagnosis of the patient is Asthma, then a Triggering Rule would set the CM of the Asthma prototype to some positive value. The Certainty Measure is raised, therefore, by rules whose action clauses suggest the prototype as a possible hypothesis.

The Certainty Measure is an indication of how closely the actual data in the case match the expected data values in the prototype. When data values are classified as Plausible Values in the prototype, the CM is raised an amount dependent upon the importance Measure of that component slot using the algorithm described in Section 5.2.2. Similarly, the Certainty Measure is lowered when parameter values are classified as Surprise Values or Possible Error Values. Another distinction between Certainty Measures and Certainty Factors, therefore, is that the importance Measures of the prototype components combine with the CM of the prototype.

Slot Name	Sample OAD Values
Certainty Measure	888
InTriggers (parameter & value)	((TLC 126) (MMF 12))
Origin (link & prototype)	(DISEASE PULMONARY-DISEASE)

FIGURE 5.5 Dynamic Information Slots and Sample OAD Values

5.2.6 Invocation Record Slots

During a consultation, a prototype is suggested either by parameter values (data triggers) or by other prototypes as possible alternate or more specific hypotheses. Two slots record this information which is used in summarizing the consultation and in explaining the system's performance. Both are used in explaining why the system is exploring a given prototype, and, in the case of data triggers, at the end of the consultation to summarize the data that strongly suggested the confirmed prototypes. The INTRIGGERS slot lists these data triggers, or (parameter value) pairs, in the format shown in Figure 5.5, thus specifying a relationship between parameters and prototypes in each consultation. There may be several inTriggers in each prototype. Figure 5.6 shows a sample of a single trigger, TLC = 126, which suggests the OAD prototype, with the resulting instantiated InTriggers slot.

SYSTEM EVENT:	EXAMPLE:
PARAMETER VALUE	TOTAL LUNG CAPACITY VALUE = 126
↓ triggers	↓ triggers
PROTOTYPE	OBSTRUCTIVE AIRWAYS DISEASE

RESULT:

OBSTRUCTIVE AIRWAYS DISEASE INTRIGGERS SLOT:

(TOTAL LUNG CAPACITY 126)

FIGURE 5.6 Instantiation of an INTRIGGERS Slot

The invocation of one prototype by another during a given consultation is recorded in the ORIGIN slot. A prototype control task can suggest that other categories of prototypes be explored, such the OAD control task, *Deduce the subtype of OAD*. The Origin slot then records the name and connecting link in the network of the suggesting prototype. Figure 5.7 shows a sample of system events resulting in an instantiated ORIGIN slot.

SYSTEM EVENT:	EXAMPLE:
PROTOTYPE	OBSTRUCTIVE AIRWAYS DISEASE
↓ suggests (link)	↓ suggests (SUBTYPE)
PROTOTYPE	ASTHMA

RESULT:

ASTHMA ORIGIN SLOT:

(SUBTYPE OBSTRUCTIVE AIRWAYS DISEASE)

FIGURE 5.7 Instantiation of the ORIGIN Slot

5.3 Production Rules

In addition to prototypes and components, the CENTAUR knowledge base includes 142 production rules: 62 Inference Rules, 14 *Summary Rules*, 42 *Refinement* Rules, 18 Triggering Rules, and 6 Fact-residual Rules. The Inference Rules, used to infer values for the prototype components, already have been discussed. The four other sets of rules are presented in the following sections.

5.3.1 Summary Rules

Those rules whose actions make summary statements about the results of the

pulmonary function tests are classified as SUMMARY RULES. A sample of a Summary Rule is shown in Figure 5.8. RULE050 is associated with the NORMAL prototype and is used to produce a summary statement about the normal pulmonary function of the patient.

```
RULE050
-------

  If:   The degree of obstructive airways disease as indicated by
           overinflation is greater than or equal to mild
  Then:  It is definite (1.0) that the following is one of the summary
            statements about this interpretation: Pulmonary Function is
            within wide limits of normal.

  PROTOTYPE:  NORMAL
```

FIGURE 5.8 A Sample Summary Rule for the NORMAL Pulmonary Function Prototype

5.3.2 Triggering Rules

Rules that refer to values of components in their premises and suggest general disease categories in their actions are classified as TRIGGERING RULES. These are antecedent rules used to "trigger" the disease prototypes. A sample Triggering Rule is given in Figure 5.9. RULE093 checks the value of one of the test measurements (the diffusing capacity for carbon monoxide, or DLCO) and suggests that this data value indicates three possible prototypes, with the certainty of this conclusion as given by the associated Certainty Measures.

```
RULE093
-------

 If:   The dlco/dlco-predicted ratio of the patient is less than 80
 Then:  1) Suggest DIFFUSION-DEFECT with a certainty measure of 900,
        2) Suggest EMPHYSEMA with a certainty measure of 800, and
        3) Suggest RLD with a certainty measure of 800
```

FIGURE 5.9 A Sample Triggering Rule

5.3.3 Fact-Residual Rules

The set of rules classified as FACT-RESIDUAL RULES is applied at a later stage in the consultation, after a set of prototypes has been confirmed. Ideally, the diagnosis, i.e., the set of confirmed prototypes, should account for all of the known facts in the case. Residual facts can be an indication that the diagnosis is not complete, or that the case is somehow exceptional. The Fact-Residual Rules attempt to make conclusions about these residual facts.

Two samples of Fact-Residual Rules for the pulmonary function interpretation task are shown in Figures 5.10 and 5.11. Rule157 is associated with the OAD prototype and is applied when OAD has been confirmed in the patient, but the Total Lung Capacity (TLC) value is not a plausible value for OAD. Thus the Total Lung Capacity must be a residual fact. Knowing that there is OAD of a sufficiently high degree, and that the value for the TLC is low for a typical OAD case, is evidence that there is a second disease process, Restrictive Lung Disease (RLD), in the patient. CENTAUR makes a conclusion statement to this effect.

The general problem of matching individual frames to a situation which in reality represents a combination of frames is complicated because the situation will not match any single frame exactly. Similarly, in medical diagnosis, the presence of multiple diseases in a patient is difficult to detect because the expected fact values for each disease in its pure form, as represented in the prototypes, may be somewhat modified. Consider, for example, the expected values for the Total Lung Capacity which are higher than normal (greater than 100) for OAD and lower than normal (less than 80) for RLD. In a mixed case of OAD and RLD in the patient, the

Total Lung Capacity value might not match the expected values for either disease prototype.[6] In this case, the inference of multiple diseases is made only late in the consultation after OAD is confirmed as matching other facts in the case. A residual fact itself can be a clue that the diagnosis is not complete.

```
RULE157
-------

  If:  1) There is not sufficient evidence for RLD,
       2) The degree of obstructive airways disease of the patient
          is greater than or equal to moderately-severe, and
       3) The tlc/tlc-predicted ratio of the patient is between 90
          and 100
  Then:  1) Mark the TLC as being accounted for by RLD, and
         2) It is definite (1.0) that the following is one of the
            conclusion statements about this interpretation: The
            reduced total lung capacity in the presence of obstruction
            indicates a restrictive component.

  PROTOTYPE:  OAD
```

FIGURE 5.10 A Fact-Residual Rule--Multiple Diseases

The second Fact-Residual Rule, RULE160 represents what Minsky has termed an "excuse", an attempt to explain an apparent misfit of fact to prototype [Minsky, 1975]. The residual fact is again the Total Lung Capacity of the patient, but in this case, the rule is associated with the *NORMAL prototype* and is *invoked* when a patient has been confirmed as having normal pulmonary function. An abnormally high Total Lung Capacity in an otherwise normal patient is an indication that the patient is "Super-Normal", and simply has exceptional lung capacity. Again, a conclusion

[6] A mixed case of OAD-RLD could have been represented explicitly as a prototype in the knowledge base. This solution in general, however, would lead to a proliferation of prototypes representing various combinations of disease in the patient, and would necessitate adding many "disease combination" prototypes to the knowledge base whenever a new disease was represented. Instead, it was decided to handle individual cases using the production rule approach described here.

statement is made to this effect. This rule also illustrates the importance of applying these pieces of expertise at a later stage of the consultation, after some determination of disease has been made. It is only *in the context* of a patient with Normal pulmonary function that the high Total Lung Capacity can be "excused". In the initial stage of processing, when prototypes are being suggested by the facts, a high Total Lung Capacity would only indicate Obstructive Airways Disease, not Super-Normal Pulmonary Function.

```
RULE160
-------

  If:  The tlc/tlc-predicted ratio of the patient is greater than or
         equal to 120
  Then:  1) Mark the TLC as being accounted for by SUPER-NORMAL, and
         2) It is definite (1.0) that the following is one of the
            conclusion statements about this interpretation: The high
            total lung capacity is consistent with super-normal
            pulmonary function.

  PROTOTYPE:  NORMAL
```

FIGURE 5.11 A Fact-Residual Rule--"Excuses" for Fact Values

5.3.4 Refinement Rules

Those rules that are used after the system has formulated lists of confirmed and disconfirmed prototypes are called REFINEMENT RULES; they are used to refine an interim diagnosis, producing a final diagnosis about pulmonary disease. Refinement Rules constitute a further set of domain expertise; they test the system's tentative conclusions, and make additional conclusions and recommendations. For example, if two diseases can account for a given pulmonary function test result and both have

been confirmed in that case, a Refinement Rule may determine which disease process should account for the test result in the final interpretation. Refinement Rules may also recommend that additional lab tests be performed. Samples of Refinement Rules are given in Figure 5.12. Rule035 is executed when an interim diagnosis indicates a Diffusion Defect. The rule tests several conditions, including the presence of OAD, to determine whether a second TLC measurement should be obtained. Rule040 is associated with the OAD prototype and attributes the cause of the patient's airway obstruction to smoking.

```
RULE035
-------

  If:  1) There is evidence for OAD,
       2) The tlc(body box)observed/predicted of the patient is not
          known, and
       3) A: The tlc(dlco)observed/predicted of the patient is less
             than 100, and
          B: The Observed-Predicted difference in RV/TLC of the patient
             is less than 20
  Then:  It is definite (1.0) that the following is one of the
         conclusion statements about this interpretation: Measuring
         TLC by a more reliable method than DLCO is necessary to
         confirm a diagnosis of emphysema.

  PROTOTYPE:  DIFFUSION-DEFECT

RULE040
-------

  If:  1) The number of pack-years smoked is greater than 0,
       2) The number of years ago that the patient quit smoking is 0,
          and
       3) The degree of smoking of the patient is greater than or equal
          to the degree of obstructive airways disease of the patient
  Then:  It is definite (1.0) that the following is one of the
         conclusion statements about this interpretation: The
         patient's airway obstruction may be caused by smoking.

  PROTOTYPE:  OAD
```

FIGURE 5.12 Sample Refinement Rules

5.3.5 Comparison to PUFF Knowledge Organization

The organization of CENTAUR's production rules differs from the organization of rules in the PUFF system in two ways. First, CENTAUR's rules are classified according to their function in the system as one of the five rule types discussed above. In PUFF, all of the rules are classified as Inference Rules, even though some of them actually summarize data or perform functions other than simply inferring information about the patient. Second, in CENTAUR rules are values of slots in prototypes, making explicit the context in which each rule is applied.

The organization of "concepts" (represented as *parameters* in PUFF and as *components* in CENTAUR) also differs. In CENTAUR there are several prototypes, each of which may contain a component slot *representing the same* concept. PUFF represents each concept by only a single parameter. The knowledge associated with the component in CENTAUR is specific to the prototype, and in general is more narrowly defined. For example, Figure 5.13 shows *plausible values* for the TLC components in three prototypes, as compared to the *expected values* for the single PUFF TLC parameter.

The PUFF parameters and their associated knowledge have been retained in CENTAUR so that the components can *inherit* information from the more general parameters when it is not already known. For example, because there are no Inference Rules associated with the TLC component in the OAD prototype, the Inference Rules associated with the more general parameter will be used to infer a value if one is needed. The more general parameters and rule lists are also used in the event that no prototype matches the given data, forcing the system to rely on

the original PUFF knowledge organization and backward-chaining control structure to perform the consultation.

Other information, such as an English translation, is associated with *parameters* instead of components since it is not particular to any specific prototype.

PUFF TLC Parameter

English Translation:	"Total Lung Capacity"
Expected Values:	Any Number
Inference Rules:	(RULE026, RULE027, ...)

CENTAUR TLC Components

	OAD Prototype	NORMAL Prototype	RLD Prototype
Plausible Values:	> 100	Between 80 and 120	< 80
Importance Measure:	4	5	5

FIGURE 5.13 Total Lung Capacity Parameter and Components

5.4 Facts

In CENTAUR, each datum that has been acquired either initially from the pulmonary function test results or later during the interpretation process is called a Fact. Each fact is represented as a frame with six slots: **Name, Value, Certainty Factor, Where Obtained, Classification in prototypes,** and **Prototype that accounts for the fact**. Facts in an EMYCIN system consist of only the first three slots of information.[7] The last three slots are used during the CENTAUR consultation

[7] In EMYCIN, the "name" slot actually consists of two parts: the parameter name itself and an *object* in the system with which that parameter is associated, for example a specific organism in MYCIN. In the pulmonary function problem there is a single object, the patient, which fills the object part of the name slot of each fact.

and by the explanation system as described below. When a fact is first introduced into the system, it's name, value, and certainty factor slots are filled in. For example, the user may specify that the Total Lung Capacity of a patient is 126 with a Certainty Factor of .8, thus creating a CENTAUR fact:

NAME: *Total Lung Capacity*, VALUE: *126*, CERTAINTY FACTOR: *.8*.

The fourth slot of a CENTAUR fact indicates where the fact value was obtained: whether from the user (this includes the initial pulmonary function test results), from the rules, or as a default value associated with a prototype component. This slot is also instantiated when a fact is created. Thus, in the Total Lung Capacity fact, the fourth slot would have the value *USER*. Information in this slot is used by the explanation system to explain where the value of the fact was obtained. It has also been used extensively in debugging the knowledge base to determine where erroneous conclusions originated.

The value of the fifth slot is a set of pairs of the form

((classification prototype) (classification prototype) ...),

where the classification is *PV*, *PEV*, or *SV* indicating that the fact is a Plausible Value, Possible Error Value, or Surprise Value in the given prototype. In the Total Lung Capacity fact, for instance, the fifth slot might contain the classification *((PV OAD) (SV NORMAL))* meaning that the value of 126 for the Total Lung Capacity of a patient would be a Plausible Value if the patient had Obstructive Airways Disease, but would be a Surprise Value if the patient were considered to have Normal pulmonary function. Appropriate pairs are added to this slot whenever fact values are classified in prototypes. For example, when the TLC=126 fact is classified as a

Plausible Value in OAD, the pair *(PV OAD)* is added to the classification slot. Information in the classification slot is used during a consultation when facts that have not been accounted for by confirmed prototypes are checked to determine what prototypes can account for them. To continue the TLC example, if TLC=126 was a residual fact, then checking this classification slot informs the system that the OAD prototype can account for the fact, and may be a reasonable prototype to explore next. The information in this slot is also used to specify inconsistencies in the match of prototypes to facts.

The last slot associated with a fact indicates which confirmed prototypes can account for a given fact value. When a *prototype* is confirmed, all of the facts that correspond to components in the prototype and *whose values are plausible* values for the component are said to be "accounted for" by that prototype. Information in this slot is used to determine which facts remain to be accounted for at any time during system processing. If the Obstructive Airways Disease prototype is confirmed as matching the data in an actual case in which the Total Lung Capacity of the patient is 126, then the Obstructive Airways Disease can account for the high Total Lung Capacity of the patient, and the last slot of the sample Total Lung Capacity fact would be filled in with the prototype name, *OAD*. If, however, the patient is confirmed to have Normal pulmonary function, the high Total Lung Capacity is not explained and becomes a residual fact. Such facts are the subject of the Fact-Residual Rules discussed earlier in this chapter. The sample Total Lung Capacity fact is shown in Figure 5.14 at three points during the consultation when slot values are obtained.

System Event	Fact Representation
[A fact is created when information is entered by the user, inferred from rules, or given as the default value of a prototype component.]	
The user is asked for the TLC measurement. The response is: 126 (.8)	Name: TLC CF: .8 Value: 126 From: USER
[The fact is classified in prototypes that have been selected as being possible hypotheses.]	
OAD and NORMAL are selected as possible hypotheses.	Name: TLC CF: .8 Value: 126 From: USER Classification: ((PV OAD) (SV NORMAL))
[As prototypes are confirmed, facts whose values are plausible values (PVs) in the prototype are marked as being accounted for by that prototype.]	
OAD is confirmed.	Name: TLC CF: .8 Value: 126 From: USER Classification: ((PV OAD) (SV NORMAL)) Accounted For: OAD

FIGURE 5.14 A Sample Fact at Three Points in the Consultation

5.5 Knowledge Representation Comparisons--PIP and INTERNIST

Two medical diagnosis systems that also use a frame-like representation for their knowledge are the Present Illness Program (PIP) and the INTERNIST system. INTERNIST diagnoses diseases in internal medicine, and with over 400 diseases represented, is currently the largest of the "AI in medicine" systems. The INTERNIST researchers have stressed diagnostic accuracy as their primary goal. PIP attempts

to simulate the behavior of an expert nephrologist in taking the history of the present illness of a patient with renal disease. The emphasis of the PIP research has been to elucidate and verify a theory of clinical cognition, and also to develop a system to help the non-expert physician provide better medical service with the aid of a computer. Figure 5.15 presents, in tabular form, a comparison of the knowledge representation structures of PIP, INTERNIST II, and CENTAUR.

In PIP, renal disorders are represented by *Hypothesis Frames* in which typical *findings* are listed, together with causally related frames, lists of findings that allow definite exclusion or confirmation of the frame, *differential diagnosis links*, and a scoring function that evaluates in numerical form the degree to which reported findings are consistent with the expected findings of the disorder.

The knowledge base of the INTERNIST system is composed of *disease entities* and *manifestations*. Each disease entity is a frame-like structure that includes a list of manifestations expected to be present with an estimate on a scale of 1 to 5 of the frequency of occurrence of the manifestation in that disease. There are also several auxiliary relations defined for each manifestation, such as the *type* of the manifestation (history, symptom, sign, etc.).

Thus in each system there are frames with typical findings or manifestations, corresponding to CENTAUR's prototype components. CENTAUR's facts are analogous to the presence of a manifestation in INTERNIST, or the presence of a finding in PIP. CENTAUR's classification of fact values as plausible values, possible error values, or *surprise values is unique*, however, and gives CENTAUR the ability to recognize and deal with erroneous or inconsistent data values. Because both INTERNIST and PIP

follow a conservative strategy that attempts to account for all data values present in each case, erroneous data values can cause both systems to continue generating progressively less sensible hypotheses until no more possible hypotheses remain.

Neither PIP nor INTERNIST use a production rule representation. All findings and manifestations are entered by the user either initially, or as the system requires further information, but are not inferred by production rules as is done in CENTAUR. Some manifestations in INTERNIST, called *Constrictors*, and some findings in PIP, serve as triggers that suggest likely hypotheses. These are similar to the CENTAUR Triggering Rules except that they do not represent *combinations of data values that* together trigger a hypothesis.

In each system there is a network or hierarchy of diseases that relates the various diseases and helps guide the formation of possible hypotheses. The hierarchies also enable the systems to produce higher-level conclusions when the data do not permit more precise judgements.

Neither PIP nor INTERNIST has an explanation system, and thus cannot explain why a given prototype is being considered, as is done in CENTAUR by means of its Invocation Record slots.

CENTAUR's emphasis on representing control information explicitly in control slots associated with each prototype is not shared by either of the two other systems. In each, there is a pre-set, general control structure that is not specific to any frame and which does not vary depending on changes in the currently most likely hypothesis. A more detailed comparison of the control structures of the three systems is given in Section 6.8.

DATA STRUCTURE	SYSTEM INTERNIST (internal medicine)	PIP (renal disease)	CENTAUR (pulmonary disease)
FRAME	Diseases (400)	Hypotheses (38)	Prototypes (24)
SLOTS	Manifestations (2000)	Findings (160)	Components (69)
	type, import, frequency	mini-frames with slots	importance, rules, default value, plausible values, possible error values
PRODUCTION RULES	(none)	(none)	Triggering Rules, Inference Rules, Refinement Rules, Summary Rules, Fact-Residual Rules (142)
DATA	present, absent, or unavailable for each manifestation	instantiated findings	Facts: value, CF, where obtained, how classified, how accounted for
FRAME HIERARCHY POINTERS	links causal, temporal	differential diagnosis, causally related hypotheses	More Specific, More General, and Alternate links
CONTROL SLOTS	(none)	(none)	To-Fill-In, If-Confirmed, If-Disconfirmed, Action
SCORING SLOTS	algorithm using import, frequency, evoking strengths	scoring function, confirmation and exclusion lists	Certainty Measure, Importance Measures
INVOCATION RECORD SLOTS	(none)	(none)	Origin (from other prototypes) InTriggers (from facts)

FIGURE 5.15 Knowledge Representation Structures in PIP, INTERNIST II, and CENTAUR

Chapter 6

Control

6.1 Introduction

This chapter discusses the representation and use of control knowledge in CENTAUR. A brief overview is given first, defining the main elements in the control scheme. A discussion of the principal stages in the consultation process and their implementation by means of higher-level, task prototypes and an agenda mechanism, follows. Comparisons of this representation of control knowledge with meta-rules [Davis and Buchanan, 1977], and with both INTERNIST II and PIP also are presented.

6.2 Control Overview

CENTAUR's basic control structure is quite simple: the system interpreter executes the top task on the agenda, and when that task is finished, the process repeats. The system terminates when the agenda is empty. A task is an action to *be taken by the system, specified as a call to a LISP predicate function.* Tasks are added to the agenda from two sources: (a) from prototype control slots, and (b) from the tasks themselves, as the execution of one task may cause others to be placed on the agenda. (See Figure 6.1.) *Tasks that are specific to a given prototype,* such as those specifying how to proceed in the consultation when a prototype is confirmed to match the data, are values of prototype control slots, as discussed in Chapter 5. *Other tasks are more general and apply to all of the prototypes,* such as the task to order a hypothesis list of prototypes. Appendix B gives a complete list

of general control tasks defined for the consultation problem. This chapter discusses the control process for a *consultation*, control for a ***prototype review*** is considerably simpler and is discussed in Section 7.4.

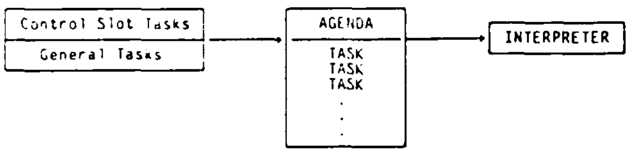

FIGURE 6.1 Control Overview

During a consultation, a prototype will be in one of three states (as illustrated by Figure 6.2): Inactive prototypes in the static knowledge base, **Potentially Relevant** prototypes that have been suggested by data values, and **Relevant** prototypes on the Hypothesis List that were selected from the set of Potentially Relevant prototypes as being possible solutions to the current task. The Hypothesis List is simply an ordered list of prototype name and Certainty Measure pairs, listed in decreasing order of the Certainty Measures.

At any one time during the consultation, there is a single **Current Prototype** which represents the system's current hypothesis about how to classify the data in the case. Two lists keep track of the prototypes that have been confirmed as matching the data in the case (the **Confirmed List**) and those that have been tested but have failed to match (the **Disconfirmed List**).

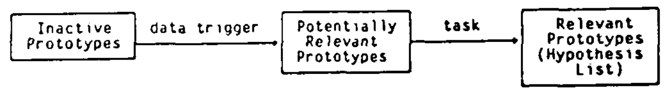

FIGURE 6.2 Prototype States

6.3 Consultation Stages

The consultation process itself proceeds in stages, listed below. **Stages are** conceptual entities useful in describing the chain of events occurring during the consultation; stages do not exist in code.[1] Many *tasks* may be executed during any one consultation stage. Explicit tasks do exist in the CONSULTATION prototype that apply sets of rules and prototype control slots in a sequence that results in the various stages.

The initial system configuration for the consultation task is shown in Figure 6.3. The CONSULTATION prototype is selected as the current prototype, and the agenda is initialized with two tasks: *Fill-in* and *Confirm* the current prototype.[2] Knowledge in the To-Fill-In and If-Confirmed control slots of the prototype directs these tasks. Filling in the CONSULTATION prototype entails asking the user for options used in running the consultation, such as a choice of strategy for exploring prototypes. The

[1] Stages are analogous to the ill-defined beginning, middle, and end games in chess.

[2] For the Prototype Review task, the initial current prototype is the REVIEW prototype, but the initial agenda tasks are the *same as* those specified here for the Consultation task.

If-Confirmed slot controls the consultation stages. Both are discussed further in Section 6.4

AGENDA
FILL-IN Current Prototype
CONFIRM Current Prototype

Current Prototype: CONSULTATION

FIGURE 6.3 Initial System Configuration for the Consultation Problem

Both prototype control slots and production rules associated with each prototype play a role in various consultation stages. The consultation process is summarized in Figure 6.4 which also indicates the roles of control slots and rules. The key stages in the consultation process are:

1) *Entering Initial Data*--Values for an initial set of parameters including some of the standard pulmonary function test results are requested.

2) *Triggering Prototypes*--Prototypes are suggested by ***Triggering Rules*** associated with parameters. Certainty Measures associated with the suggested prototypes are increased.

3) *Scoring and Selecting a Current Prototype*--The hypothesis list of suggested prototypes is ordered by Certainty Measures so that the first prototype represents the system's best hypothesis about how to match the data in the case.

4) *Filling in Prototype*--The prototype components are filled in with facts already determined in the case. New facts may also be inferred, and if new prototypes are suggested, the consultation

returns to stage 2. This instantiation is guided by tasks in the ***To-Fill-In Slot*** of the prototype. These tasks, in turn, guide the invocation of the ***Object-Level Inference Rules*** which infer values for prototype components. Questions are asked of the user at this stage when rules fail to infer the needed information, or when the components are of the *ASKFIRST* variety discussed in Section 2.3.

5) *Testing Match*--The prototype is tested to see whether there is a close enough match of actual fact values in the case to expected values in the prototype to confirm the prototype. Tasks in the ***If-Confirmed Slot*** or the ***If-Disconfirmed Slot*** are added to the agenda accordingly. These tasks generally suggest further sets of prototypes to be explored, and the consultation returns to stage 3.

6) *Accounting for Data*--When there are no more prototypes to be explored, those facts matching expected values in the confirmed prototypes are marked "accounted for" by the prototypes. The ***Fact-Residual Rules*** are applied to any remaining facts as a set of expertise that attempts to explain these apparent discrepancies.

7) *Refining Diagnosis*--The ***Refinement Rules*** are applied to this interim diagnosis to produce a final diagnosis of pulmonary disease in the patient.

8) *Summarizing Results*--The ***Summary Rules*** associated with the confirmed prototypes are applied.

9) *Printing Results*--Tasks in the ***Action Slot*** of each confirmed prototype that control the printing of final results are added to the agenda and subsequently executed.

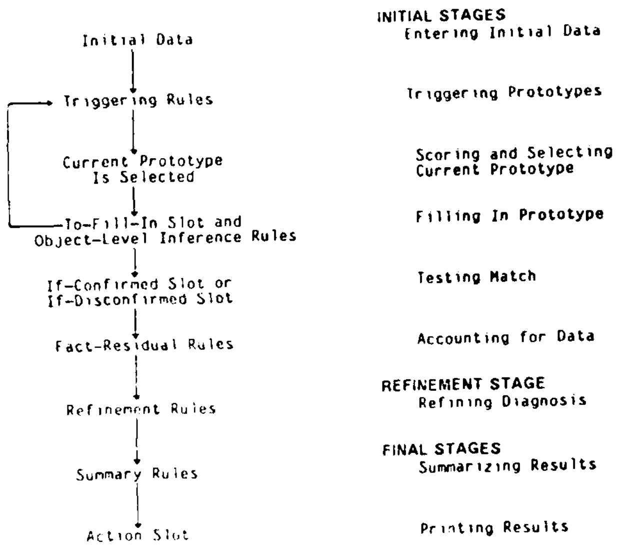

FIGURE 6.4 Overview of the Consultation Process

(The control tasks that specify the application of prototype control slots and rules in the sequence listed above are themselves tasks in the CONSULTATION prototype shown in Figure 6.5.)

6.4 Higher-Level Prototypes

Prototypes are also used in CENTAUR to represent three high-level tasks: the consultation task (CONSULTATION prototype), the prototype knowledge review task (REVIEW prototype), and the general pulmonary function interpretation task (PULMONARY-DISEASE prototype), which itself is a kind of consultation. These prototypes share the same set of possible slots as the pulmonary disease prototypes discussed in Chapter 5. This uniform encoding of knowledge provides the advantage that auxiliary functions developed for the disease prototypes, including those dealing with explanation and knowledge acquisition, can be used for these higher-level prototypes as well. The most important slots for these prototypes are the control slots which represent the steps to be done in performing the given task. No component slots have been defined for these prototypes; components are principally used in matching prototypes to data, and in this case, the user specifies the task desired, making a match unnecessary. The other slots that have been used for these prototypes are slots for bookkeeping information, slots for English phrases used in communicating with the user, slots that point to other prototypes in the *prototype network*, and in the PULMONARY-DISEASE prototype, a slot for the Refinement Rules applicable to the general problem of pulmonary function interpretation.

In the CONSULTATION prototype in Figure 6.5, the To-Fill-In slot contains three tasks that set variables for the consultation: a tracing level (from 0 to 3) that allows the user to select the level of program tracing, a variable that specifies whether tasks will be printed as they are added to the agenda and as they are chosen to be

executed, and a *variable that sets a strategy* for selecting the current best prototype from the hypothesis list. The user can choose one of three strategies: a *confirmation strategy*, which selects the prototype that is the best match to the data and attempts to confirm that prototype, an *elimination strategy*, which selects the prototype that is the worst match to the data and attempts to eliminate that prototype, and a *fixed-order strategy*, which always explores prototypes in a pre-set order.

The first three tasks in the If-Confirmed slot set other control variables for the consultation. First, the confirmation threshold is set here to 0; that is, if the Match Measure of the prototype is above 0, it will be confirmed. Second, the percentage *of slots that must have values before the prototype* is tested for confirmation is set to .75. In other words, at least three-fourths of the slots must have values before the prototype Match Measure is computed and tested. Third, the default procedure for filling in prototype slots is set to fill in slots in decreasing order of their Importance Measures, so that the most important information is determined first. This is used only when there is no To-Fill-In slot specified for the prototype (as in the OAD prototype in Figure 5.3), or when there are slots remaining to be filled after the To-Fill-In tasks have been executed.

Unlike the variables in the To-Fill-In slot of the CONSULTATION prototype which are set by the user, variables in the If-Confirmed slot represent control choices that *are less likely to be modified* for an individual consultation, and thus are set within the prototype. They deal with control at the system design level and represent decisions with which the typical user would not be concerned. Representing them

here makes these control choices explicit for the system designer, and facilitates making changes to the control structure and experimenting with different variations of the basic control scheme, such as trying different confirmation thresholds. This explicit representation allows the control knowledge to be explained by the system, which in turn helps domain experts to become more familiar with CENTAUR's control structure.

The remaining tasks in the If-Confirmed slot control the stages in the consultation process. The first, that of determining the domain of the consultation, defaults to the pulmonary function domain, but allows the representation of other domains, such as infectious diseases, for future expansion of the knowledge base. Execution of this task suggests the PULMONARY-DISEASE prototype, which then is selected as the current prototype. Filling in, confirming, and accounting for facts complete the initial hypothesis-formation stages of the consultation. At this point, CENTAUR has formed its interim diagnosis *or* overall hypothesis about the interpretation of data in the case. The final tasks are the refinement, summary, and action stages of the consultation.

The REVIEW prototype in Figure 6.6 has two control slots, a To-Fill-In slot, which asks for the prototype to be reviewed, and an If-Confirmed slot, which specifies the information to be reviewed. Thus a system designer can specify what information he wishes to use, and the order in which to present it by modifying control knowledge in these slots.

The PULMONARY-DISEASE prototype in Figure 6.7 represents knowledge common to all of the pulmonary disease prototypes, such as the set of general

Refinement Rules, and also specifies control knowledge for the pulmonary function interpretation task. The more specific pulmonary disease prototypes *inherit* knowledge from this more general prototype. The To-Fill-In slot requests initial pulmonary function test results, which in turn suggest more specific disease prototypes. The If-Confirmed slot then specifies that these more specific disease prototypes be explored. The Action slot controls the printing of the final interpretation.

```
CONSULTATION
------------
NAME:  CONSULTATION
HYPOTHESIS:   A consultation is desired.
EXPLANATION:   Consultation
AUTHOR:  AIKINS
DATE:  27-OCT-77 15:58:45
MORESPECIFIC:  (DOMAIN PULMONARY-DISEASE)

TO-FILL-IN:
   Ask for the TRACING-LEVEL for the CONSULTATION
   Ask for the AGENDA-PRINTING for the CONSULTATION
   Ask for the STRATEGY for the CONSULTATION

IF-CONFIRMED:
   Set the confirmation threshold to 0
   Set the percentage of filled-in slots necessary
        to confirm the prototype to .75
   Set the default procedure for filling in slots
        to fill in slots in decreasing order of their
        Importance Measures
   Determine the domain of the consultation
   Select the current best prototype
   Fill in the prototype
   Apply tasks in the If-confirmed slot
   Mark facts that are accounted for by the prototype
   Apply the refinement rules associated with the
        confirmed prototypes
   Apply the summary rules associated with the
        confirmed prototypes
   Execute actions associated with the confirmed
        prototypes
```

FIGURE 6.5 The CONSULTATION Prototype

(Putting To-Fill-In and If-Confirmed slots on the agenda as tasks associated with the CONSULTATION prototype causes the system to ask for values, set thresholds, determine the domain of the application, and invoke sets of rules and prototype control slots during the consultation.)


```
IEW
-----
NAME:  REVIEW
HYPOTHESIS:   A review of prototype knowledge
              is desired.
EXPLANATION:   Review
AUTHOR:  AIKINS
DATE:  15-Nov-79 14:06:37

TO-FILL-IN:
   Ask for the name of the prototype to be reviewed

IF-CONFIRMED:
   Review the components, their expected values, and
        their importance measures
   Review knowledge in the control slots
```

FIGURE 6.6 The REVIEW Prototype

```
PULMONARY-DISEASE
-----------------
NAME:  PULMONARY-DISEASE
HYPOTHESIS:   An interpretation of the pulmonary
              function tests is desired.
EXPLANATION:   Pulmonary Disease
AUTHOR:  AIKINS
DATE:  27-OCT-77 16:51:39
MOREGENERAL:  (TASK CONSULTATION)
MORESPECIFIC:  (DISEASE OAD) (DISEASE RLD) (DISEASE
                DIFFUSION-DEFECT) (DISEASE NMD)
                (DISEASE NORMAL) (DISEASE SUPER-NORMAL)

TO-FILL-IN:
   Enter the results of the pulmonary function tests

IF-CONFIRMED:
   Determine the DISEASE of PULMONARY-DISEASE

ACTION:
   Summarize the data given in the prototypes
   Display the findings about the HFP diffusing capacity
   Display the findings about the pulmonary function tests
   Display the conclusion statements about this
        interpretation
   Display the summary statements about this
        interpretation

REFINEMENT-RULES:  RULE087 RULE091 RULE092 RULE141
```

FIGURE 6.7 The PULMONARY-DISEASE Prototype

6.5 The Agenda of Tasks

CENTAUR maintains an Agenda of tasks to be performed during a consultation. Each task entry includes a Source for the task, and a **Reason** that the task was added to the agenda as in AM (Lenat, 1976). (See Figure 6.8.) Tasks are added to the agenda from prototype control slots and as a result of executing other tasks. Tasks are removed from the agenda only as a result of executing them. In the sample figure, the first task shown, *order the hypothesis list*, has been placed on the agenda as a result of executing another task that added new prototypes to the hypothesis list. The second task is an example of a task from a prototype control slot.

The tasks are executed in a last-in, first-out (LIFO) order; thus the agenda operates as a stack in which tasks are always added to the top, and the top task is the one executed. Each set of tasks in the prototype control slots are added to the agenda as a group, thus preserving the order in which they have been specified. This is important because the knowledge in the prototype control slots specifies not only the tasks to be performed at specific times during the consultation, but also the order in which those tasks should be performed. For example, control tasks in the CONSULTATION prototype must be executed in the order specified for the consultation to progress through its various stages. The execution of one task may cause others to be added to the agenda, thus expanding the top task into other, more detailed tasks. Operating the agenda as a stack thus has the effect of moving depth-first through the prototype network as more and more specific prototypes are explored.

The reasons associated with prototype control tasks are generated from the name of the prototype and the name of the control slot where the task originated. The reasons associated with each general task are text strings stored with that task which briefly explain what the system is doing.

The sources and reasons for tasks can be shown to the user in two different ways. During a consultation, a full tracing of the tasks being executed can be shown. This includes an English translation of the task and its source, as well as the reason for the task, which is useful for understanding the flow of control. A more terse mode of printout that includes only the task name and source name can also be given for those who are more familiar with the system, and has been particularly useful during system development. The user can also type a special control character at any time during the consultation, and the system will print the next task to be executed and its source and reason in either regular or terse mode.

```
TASK: Order the Hypothesis List
SOURCE: Task adding new prototypes to the Hypothesis List
REASON: Because new prototypes have been added to the
   Hypothesis List, it should be checked to see that it is
   ordered according to which prototype best fits the facts.

TASK: Deduce the degree of OAD
SOURCE: OAD If-Confirmed Control Slot
REASON: Because Obstructive Airways Disease has been
   confirmed, an attempt should be made to deduce the
   degree of OAD.
```

FIGURE 6.8 Sample Agenda Entries

6.5.1 Advantages of the Agenda Mechanism

One motivation for placing tasks on an agenda is that it allows the system to "look ahead" to see what tasks are remaining to be executed. This is used, for example, in deciding when to apply the Fact-Residual rules, in order to see if there are prototypes remaining to be explored that could account for residual facts. Thus the state of the agenda at any time shows exactly which control tasks remain to be executed in the consultation.

Because much of the control for the consultation is defined in prototypes, the [illegible] of the program will be different for different sets of data matching different prototypes, as general system tasks and prototype-specific tasks are [illegible] in executing the consultation. The agenda provides the means for [illegible] of the tasks and the reasons for considering them.

Another advantage of the agenda scheme is that it forces key steps in the interpretation process to be defined as single tasks, resulting in a highly-modular, [illegible] system. [illegible] key steps in the interpretation process, in turn, [illegible] what the system is doing at any time.

Defining distinct tasks for the prototypes also allows the system to deal with a task that fails. Although some tasks, such as ordering the hypothesis list, are merely executed, others [illegible] by the prototypes may succeed or fail, depending on the data [illegible]. For instance, given the task, "determine the degree of [illegible]," the task will [illegible] if no degree prototype is a close enough match to the data. If a prototype task succeeds, then a statement is printed to that effect. If it fails, the

user is given the option of discontinuing the consultation.[3] Figure 6.8 shows a portion of two consultations to illustrate both situations. A task failure may indicate that there is a serious data error or a data base error that will invalidate the conclusions. Thus, by indicating task failures to the user as soon as they occur, the time it would have taken to perform the consultation is not wasted.

```
The task to determine the SUBTYPE of OAD
   succeeded.
The solutions were:   ASTHMA, BRONCHITIS
    .
    .
    .
-------------------------------------------

The task to determine the SUBTYPE of OAD
   failed.
Do you wish to continue the consultation anyway?
** YES
    .
    .
    .
```

FIGURE 6.9 Notifying the User of Task Status

The ability of a system to know when it has failed is especially critical for expert systems that give advice to users. Failures can occur either because (a) known information does not support any hypothesis strongly enough to outweigh disconfirming information, or (b) there is not enough information known upon which to base a decision. In CENTAUR, a prototype Certainty Measure which falls below the confirmation threshold is a failure of the first type. Similarly, in EMYCIN systems, a rule will fail if the minimum certainty of its premise clauses falls below .2.

[3] If the program is being run in a "batch mode" where the data are supplied by a tape and there is no user, then the default is to continue the consultation.

[illegible] features of the second type because the prototype slots specify what information must be known in each situation, so at any time, the extent [illegible] is evident from the number of slots that have values. Matching of a prototype requires that a certain percentage of the slots have values as well as that the Certainty Measure be above threshold. That is, CENTAUR uses its knowledge of prototypical situations to judge the *extent* of its knowledge, as well as the closeness of the match. This is not the case in rule-based systems in which each rule makes a conclusion based solely on the closeness of the match of known information to the conditions in the rule premise. Consider the following example as it applies to both systems. In the pulmonary function problem, there are five indicators for the degree of OAD. In CENTAUR these are represented as slots of the OAD degree prototypes. In PUFF, they are clinical parameters operated on by five sets of rules which make five different conclusions about the degree of OAD. These conclusions are then combined into one overall degree as has been discussed in Section 3.6. If there is missing information about one, or even two of the degree indicators, each system still should be able to reach a conclusion on the basis of the information that is known. If there is missing information about three or more of them, however, each system at least should be aware that conclusions were formed on the basis of only partial evidence. That is, the system should know that it may not know enough to reach a valid conclusion. This is the case in CENTAUR where the task of determining the degree of OAD would fail because not enough information is known. The user then is given the option of continuing, with conclusions based on only partial data. In PUFF, however, conclusions are made about the two known degree indicators, and a conclusion of an overall degree based only on those two indicators is derived, with no indication to the user of the partial basis for the decision.

The ability to reach conclusions on the basis of partial information is a desirable feature of expert systems where complete information often may not be known. In the absence of *situation-specific* expectations of information that should be known, however, such systems can not properly judge the validity of their own conclusions.

6.5.2 Comparison to Other Agenda Systems

Many other systems have used agendas as part of their control structures, for example, AM [Lenat, 1976], DENDRAL [Lindsay, et al., 1980], KRL [Bobrow and Winograd, 1977], and GUS [Bobrow, et al., 1977]. In AM, the agenda was used principally to manage a huge task-selection problem. At any time there were many *plausible tasks to consider*, so the tasks with the strongest reasons for being executed would be chosen. Tasks were selected on the basis of a computed priority. CENTAUR does not have a large task-selection problem, and will, in fact, execute all of the tasks on its agenda. Rather, the primary purpose of the agenda in CENTAUR is to allow an easily accessible and explicit representation of the control tasks. The reasons associated with tasks in AM's agenda were useful in computing scores *in order to determine the top-priority* task. In CENTAUR, the sources and reasons are defined for purposes of understanding system performance.

The agendas in GUS and KRL are used as part of the central control process, but not to explain reasoning as is done in CENTAUR. In GUS, the agenda is used to decide what should be done next. The system puts potential processes on the agenda, and then operates in a cycle in which it examines this agenda, chooses the next job to be done, and does it. In KRL, the agenda is a priority ordered list of

queues, with all processes on a higher priority queue run before any process on a lower priority queue.

The agenda in [illegible] is used by the "predictor" to keep track of fragment ions waiting to be processed by the rule set. Initially the agenda contains only the unfragmented molecular ion. As a result of fragmentations to this ion and to the resulting fragmented ions, new ions are created and added to the agenda. For each ion, all rules that match anywhere in the ion are applied. The use of the agenda allows a breadth-first behavior in which the primary ion fragmentations are operated on first. A history of where each ion originated *(who placed it on the agenda)* is saved and printed in a summary; however, no interactive explanation is available.

5.6 Advantages of CENTAUR's Control Representation

Associating control knowledge with prototypes in CENTAUR allows domain experts to specify a different set of control tasks for each prototypical situation. In the pulmonary domain, for example, the expert asks different consultation questions when OAD has been confirmed rather than some other disease. Further, because this control knowledge is separate from inference rules, the expert does not have to anticipate and correct incidental interactions between control and inference knowledge.

Representing the entire consultation process itself as a *prototype* leads to even more significant advantages. First, the system designer's conception of a consultation process is clearly defined for all system users. Second, representing each consultation stage as a separate control task allows entire stages to be added

or removed from the consultation process. For example, the *Refinement Stage*, which uses additional expertise to improve upon an interim conclusion, was omitted during early stages of system development for the pulmonary function problem. Third, "filling in" a consultation prototype with user-specified options, such as a choice of *strategy* for choosing the current best prototype (confirmation, elimination, or fixed-order), results in a control structure that can be tailored to the desires of each individual user.

The organization of knowledge into prototypical situations allows the user to *more easily identify the affected set of knowledge* when changes to the knowledge base are desired. Points at which specific control knowledge is used during the consultation are clearly defined, with the result that it is easier to predict the effects of any modifications that are made. Four different points during the consultation required prototype-specific control knowledge; more slots, for example a slot that associates a matching criterion specifically with each prototype, could be added. The system designer also has the flexibility to define any new control tasks as they are required.

Explicit representation of control knowledge also facilitates explanations about that knowledge. In addition to the HOW and WHY keywords available in EMYCIN, a new keyword, CONTROL, has been defined, so that a user of the system can inquire about the control task motivating a current line of reasoning. (See Section 7.2.1.4.)

These control slots can be viewed as attached procedures, attached to the prototype itself, instead of to the slot as in NUDGE [Goldstein and Roberts, 1977]

[illegible] in 5.5 [Bobrow et al. 1977]. The main difference is that the control tasks in the prototype are not simply executed as attached procedures, but are instead added to the system's agenda, and thus are intermingled with other tasks. A second difference between tasks in the control slots and a general attached procedure is that the control tasks are specified in the same constrained format as are the premise and action clauses of the production rules so that they can be examined by the system for explanation and other purposes.

6.7 The Consultation Process Expanded

In Section 6.3, the main stages in the consultation process were listed with a brief description of each. This section combines the previous discussion of those stages, the prototype control slots, and the agenda mechanism into a stage-by-stage summary of the consultation process.

6.7.1 The Initial Stages of the Consultation

6.7.1.1 Selecting a Current Prototype

Recall that *potentially relevant* prototypes in CENTAUR are those that have been triggered by one of the Triggering Rules, that is, antecedent rules associated with parameters. (See Figure 5.9.) When a prototype control task is executed during the consultation, such as the task of determining the degree of OAD, all potentially relevant prototypes that also have a "degree" relationship to the OAD prototype are placed on the Hypothesis List of prototypes actively being considered as possible

matches to the data. These are the *relevant* prototypes in Figure 6.2. In this case, they are relevant to the task of determining the degree of OAD. The set of relevant prototypes is further constrained by selecting only those that also were suggested by the data. If there are no OAD degree prototypes suggested by the initial data, then *all of the OAD degree prototypes are added to the* Hypothesis List. The *consultation strategy* chosen by the user then determines which prototype will be selected as the *Current Prototype*.

Figure 6.10 illustrates the three different consultation strategies that are implemented in the system. A single set of patient data was run through the system following each of the strategies. The *confirmation strategy* selects the prototype that is currently the best match to the data (as indicated by its Certainty Measure) and attempts to confirm that prototype; the *elimination strategy* selects the prototype that is currently the worst match to the data and attempts to eliminate that prototype from consideration, and the *fixed-order strategy* always attempts to match prototypes in a pre-set order. An intermediate level of tracing was chosen to show the actual hypothesis list under consideration at the point in the consultation where an OAD degree prototype was to be selected as the Current Prototype.

Confirmation Strategy

```
                    (Hypothesis List is presented in decreasing
                       order of prototype Certainty Measures)

Hypothesis List:  (SEVERE-OAD 756) (MILD-OAD -885)
     (MODERATE-OAD -886) (MODERATELY-SEVERE-OAD -929)

CURRENT PROTOTYPE:  SEVERE-OAD                (System selects
                                              "best" prototype)

I am testing the hypothesis that there is
     Severe Obstructive Airways Disease.
-------------------------------------------------------------
```

Elimination Strategy

```
Hypothesis List:  (SEVERE-OAD 756) (MILD-OAD -885)
     (MODERATE-OAD -886) (MODERATELY-SEVERE-OAD -929)

CURRENT PROTOTYPE:  MODERATELY-SEVERE-OAD     (System selects
                                              "worst" prototype)

I am testing the hypothesis that there is
     Moderately-Severe Obstructive Airways Disease.
-------------------------------------------------------------
```

Fixed-order Strategy

```
Hypothesis List:  (SEVERE-OAD 756) (MILD-OAD -885)
     (MODERATE-OAD -886) (MODERATELY-SEVERE-OAD -929)

CURRENT PROTOTYPE:  MILD-OAD     (System selects prototypes in
                                  a pre-set order, in this case,
                                      by increasing degree)

I am testing the hypothesis that there is
     Mild Obstructive Airways Disease.
```

FIGURE 6.10 Three Consultation Strategies for Choosing the Current Prototype

6.7.1.2 Processing the Current Prototype

Processing of the Current Prototype, as specified by control knowledge in the CONSULTATION prototype, involves two steps: instantiating prototype slots with values, and evaluating whether there is a close enough match between these data values and the prototype's set of plausible data values in order for the prototype to be confirmed. Knowledge about instantiating the prototype is contained in the To-Fill-In control slot, and generally involves determining values for each of the components in a specific order.[4] If the prototype has no To-Fill-In slot, the default procedure for filling in slots which is specified in the CONSULTATION prototype (see Section 6.4) is used.

When the tasks in the To-Fill-In slot have been executed, and the percentage of component slots that have values is above the necessary percentage specified in the CONSULTATION prototype, a Match Measure is computed using the algorithm described in Section 5.2.2 to test the match of the prototype to the data. The result of this test is either to confirm or disconfirm the prototype.

6.7.1.3 Selecting the Next Current Prototype

Once the current prototype is confirmed or disconfirmed, knowledge in the IF-CONFIRMED or IF-DISCONFIRMED slot specifies tasks that the system should perform next. This may include tasks that help guide the selection of the next current prototype. For example, in the OBSTRUCTIVE AIRWAYS DISEASE prototype in Figure

[4] Note that UNKNOWN is an acceptable value in the system. A value will be UNKNOWN if the user gives the response UNKNOWN when asked for the value, and rules associated with the component fail to infer a value.

5.3, the IF-CONFIRMED slot specifies that a Degree prototype and a Subtype prototype be determined for OAD. The relationships between prototypes, for example, the "Degree" relationship between OAD and MILD-OAD, are the links between the prototypes in the prototype network. All prototypes with the specified relationship are potential solutions to the task. Of these, the system selects the subset of prototypes that already have been triggered by the data, if there are any. If there are none, then all of the prototypes with the specified relationship are placed on the Hypothesis List. Components of those prototypes for which fact values already are known are instantiated, thus setting their Certainty Measures. The Hypothesis List is then ordered, and the system selects the new Current Prototype.

6.7.1.4 Accounting For Facts

When a prototype is confirmed that has no IF-CONFIRMED slot or that specifies no further set of prototypes to be considered,[5] the system marks all facts that can be accounted for by one of the confirmed prototypes. These are facts whose values are represented in a prototype as Plausible Values. If more than one prototype can account for the fact, the most specific prototype is selected. All remaining facts not thus accounted for form a pool of residual facts. Fact-Residual Rules are applied to these facts as a set of expertise that attempts to explain each apparent discrepancy on the basis of other conclusions already made in the case.

[5] Prototypes that do not specify further prototypes to be considered correspond to those that are most specific in a general to specific hierarchy of prototypes.

6.7.1.5 Deciding When to Stop

If no facts remain to be accounted for from the previous step, the system determines that it has completed the initial stages of the consultation. The Confirmed List of prototypes then represents the system's hypothesis about how to classify the data in the case. If facts remain to be accounted for, the system inspects previously-saved Hypothesis Lists to determine if some of those prototypes can account for the facts. If so, the consultation returns to its earlier stages, selecting one of those prototypes as the Current Prototype in an effort to determine whether there are multiple diseases in the patient. If there are no prototypes that can account for such remaining facts, then the system continues to the next stage anyway, but notes that there are unexplained data, and incorporates this information into the final interpretation presented to the user.

6.7.1.6 The Initial Stages of the Pulmonary Function Problem

For the pulmonary function problem, the general PULMONARY-DISEASE prototype is selected to be the first Current Prototype. Its To-Fill-In slot specifies that the most important pulmonary function test results be entered as initial data values. These data values in turn trigger the various disease prototypes. The PULMONARY-DISEASE prototype has no components and will always be confirmed by the general matching criteria. Its If-Confirmed slot then specifies that the system will select the highest ranking disease prototype (in a confirmation strategy, for instance) as the next Current Prototype. The control tasks specified in the To-Fill-In slot for this prototype are then executed and the process reiterates.

6.7.2 The Refinement Stage

In the Refinement stage, additional knowledge is used to revise the final recommendations to the user. This knowledge is represented in Refinement Rules associated with the confirmed prototypes. The Refinement Rules constitute a further set of domain expertise that considers the system's tentative conclusions and may modify these conclusions in some way. For example, if two indicated diseases can not both be present in the patient simultaneously, a Refinement Rule may choose between them. [illegible] may be sought at this stage, e.g., further lab tests may be suggested or additional test results may even be required before a [illegible]. The result of execution of the Refinement Rules is a final [illegible] and a list of all facts in the case together with an [illegible] for which facts

[illegible]

[illegible] the Summary Rules[7] associated with the [illegible] the tasks specified in the ACTION slot of each [illegible]. Typically, these tasks print conclusions [illegible]. The tasks in the ACTION slot of the PULMONARY-[illegible] interpretation and pulmonary diagnosis

[illegible] order used for executing these rules. It [illegible] to the lists of Refinement Rules to effect some [illegible] Refinement Rules associated with the confirmed [illegible] there are already so few rules being tried that no [illegible] strategy has been necessary.

[7] Recall that the premise of a Summary Rule typically checks the values for one or more parameters and the action generates an appropriate summarizing statement.

6.8 Related Research

This sections compares CENTAUR's representation of control knowledge with that of other systems. It first examines the idea of using *meta-rules* to guide the invocation of *object-level inference rules, and* contrasts this kind of control knowledge with that represented in CENTAUR. It then examines INTERNIST II and PIP and compares control in these systems with that in CENTAUR.

6.8.1 Meta-Level Knowledge

The representation of *meta-level* knowledge as meta-rules for rule-based systems was presented by Davis in TEIRESIAS [Davis, 1976], a system designed to function as an assistant in the construction of high performance programs. The meta-rules in TEIRESIAS operated on a set of object-level inference rules, and made conclusions about the utility of those rules in given situations. Thus they represented strategies for using object-level knowledge: either which rules would definitely not be useful in a particular situation, or which rules should be tried before others to infer new information.

The arguments for a meta-rule consist of (a) a system goal, such as *determine whether the Obstructive Airways Disease is reversible*, and (b) a list of object-level rules, such as those used to determine reversibility of disease. The result of executing the meta-rule is a reordered or possibly pruned list of the rules that can be used to satisfy the goal in that particular case. Thus one meta-rule in PUFF might be *If Obstructive Airways Disease is being explored, then use only those object-level inference rules that mention Obstructive Airways Disease in their premise clauses.*

The effect of this meta-rule would be to eliminate from consideration all object-level rules dealing with reversibility of disease but not relevant to the reversibility of OAD in particular.

CENTAUR guides the invocation of object-level inference rules by associating them explicitly with the context in which the rules are applied. For example, the list of rules to determine the reversibility of Obstructive Airways Disease is associated with the REVERSIBILITY slot in the OAD prototype, and is already constrained to include only those object-level inference rules that apply to OAD.

Both prototypes and meta-rules attempt to *focus attention* on the most relevant rules. They provide a means to retrieve those rules: prototypes organize rules by situations and consultation stages in which rules are applied, and meta-rules reference rules indirectly according to the content of the rule. It is conceivable that meta-rules could be used to retrieve object-level rules according to situation or consultation stage *if that knowledge were explicit in the content of the object-level rule*; however, as discussed in Chapter 3, this is not the case in either MYCIN or PUFF. Prototypes represent some knowledge, such as situations and stages, that is not explicit in MYCIN or PUFF rules.

Prototypes represent other forms of meta-level knowledge as well. Importance Measures associated with the component slots represent meta-knowledge about the relative importance of each slot. The TLC slot, for example, would be filled in before the REVERSIBILITY slot because the Importance Measure for the TLC slot is much higher. Other prototype slots represent knowledge about the prototype itself, such as those specifying the relationship of one prototype to others in the knowledge base.

Control knowledge represented in prototype slots is another type of meta-knowledge which specifies the next goal for the system. Meta-rules have been applied as strategies to *satisfy* system goals, but thus far have not been used to *specify* the goals themselves.

In summary, meta-rules can express some of the same strategies that have been represented by associating rules with prototypes, but meta-rules depend on the content of object-level rules. Prototypes focus attention on the most relevant object-level rules as defined by the current situation and stage of the consultation. Prototypes also represent other forms of meta-knowledge not represented by meta-rules. Further, meta-rules, in a system without prototypes, are subject to the same criticisms due to their implicit representation of knowledge, that have been discussed for object-level rules. Both prototypes and meta-rules could be used within the same system, with prototypes representing explicit contexts, stages, and control knowledge, and meta-rules dynamically reordering lists of rules associated with the prototype components.

6.8.2 INTERNIST and PIP

Knowledge representation comparisons for INTERNIST II, PIP, and CENTAUR were given in Section 5.5. This section compares the control structures and the representations of control knowledge in these systems. Figure 6.11 lists the main control steps and summarizes how they are represented in each system.

In INTERNIST key manifestations (findings, signs, or symptoms of the disease) called **constrictors** suggest diseases with a certain *evoking strength*. Each disease

is associated with a list of these *manifestations*, and each manifestation in the list has a number estimating its *frequency* of occurrence in that disease. Related diseases are then combined to form an overall diagnosis (using what is termed a *multiple problem generator*), so that a scoring algorithm can take into account the *relationships between diseases*. *The program* uses its scoring algorithm to choose the single disease that is most likely considering the findings in the case. The other diseases are partitioned into two sets: those that are *complementary* to the chosen disease in the sense that the two together can account for more of the findings than either alone, and those *that are competing* with the chosen disease as being likely matches to the data in the case. A questioning strategy is then selected (one of RULE-OUT, NARROW, DISCRIMINATE, or PURSUE) depending on the number of competing diseases, and the type of information required. Diseases are confirmed when their scores pass a numerical threshold. This cycle of scoring, disease selection, partitioning and question strategy selection is repeated after each set of findings is entered. The program continues until all of the known manifestations are accounted for by some confirmed disease.

CENTAUR and INTERNIST both use knowledge of prototypical cases to guide processing, but in CENTAUR, *control knowledge* used in the consultation is also represented in the prototypes. In INTERNIST there is no explicit representation of control processes that would allow the system to explain its reasoning. Again, the emphasis of INTERNIST was not on explanation or knowledge acquisition, but rather on performance over a very broad area of medicine. CENTAUR, on the other hand, deals with a much smaller, more constrained medical area. The emphasis is not as much on performance as it is on the explicitness and flexibility of the representation.

In PIP, findings reported to the program by the user are matched against expected findings in each *hypothesis frame*. Some findings are classified as *triggers*. If a finding matches one of the triggers of a hypothesis, that hypothesis is immediately *activated*. *Inactive* hypotheses are those that have not been suggested by findings or those that have been considered and rejected in light of the available evidence. A hypothesis may also be *semi-active* meaning that it is not actively under consideration, but a closely related, complementary hypothesis has been activated. An *estimate of likelihood* for each frame is computed by combining a scoring function, which measures the fit of observed findings to expected findings in the frame, with the ratio of the number of findings accounted for by the frame to the total number of reported findings. Confirmation occurs when this likelihood estimate rises above a certain threshold, or when a finding sufficient to confirm the frame is reported. Similarly, a frame is inactivated if the likelihood estimate is sufficiently low, or if a finding sufficient to exclude the frame is reported.

The algorithm used by PIP to select questions to ask is as follows: When a *finding* is introduced, PIP re-evaluates all of the affected hypothesis frames, identifies the highest-scoring active hypothesis, and chooses *one of its expected* findings to ask about. If all of its expected findings have already been investigated, then PIP pursues expected findings of hypotheses complementary to the leading one. PIP continues to ask questions of the user until there are no more unanswered questions in the active hypotheses or *any of their causal relatives*.

PIP also uses knowledge of prototypical cases to direct the consultation. Some control knowledge *in PIP is represented* in the frame itself, for example, the lists of

findings *specified* as capable of confirming or excluding the frame, and the scoring function that is used to compute the likelihood estimate. There is, however, no explicit representation of the main control algorithm specified in the preceding paragraph which results in questions being asked of the user.

Both PIP and INTERNIST include a "triggering" process, but neither uses a production rule representation for these triggers. In INTERNIST, certain manifestations are triggers for diseases, and in PIP triggers are specific findings in the frame. Representing this knowledge as rules in CENTAUR allows users to add new rules when they are needed without necessitating modifications to the existing knowledge structures. Further, because each rule represents a complete piece of knowledge (as opposed to being a part of another knowledge structure), the English translation of each rule is useful in explaining the knowledge about typical cases expressed in that rule.

Fact-Residual Rules, which in CENTAUR attempt to explain remaining facts or redirect processing in light of those remaining facts, are not used in *PIP* or INTERNIST. INTERNIST's termination criterion stops the consultation only when all manifestations have been associated with a confirmed disease. This sometimes causes the system to try progressively less likely diseases in order to account for a manifestation that could be an anomaly, or even an error. PIP similarly terminates only when there are no more findings to ask about for any of the active hypotheses or their causal relatives. Specific rules applied to the remaining manifestations in INTERNIST or findings in *PIP* would allow *each system to terminate before* questioning moves *too far afield*. *Similarly*, neither system has Refinement or its Summary Rules,

that is, separate sets of domain expertise applied after an interim diagnosis has been reached.

The major difference in CENTAUR and the other two systems is that CENTAUR's control structure is centered around the concept of typical cases and typical consultations explicit in its disease and CONSULTATION prototypes. One could conceive of a *CONSULTATION prototype for each of the other systems*. In INTERNIST, for example, a CONS' TATION prototype would make explicit the control cycle of *scoring*, *disease selection*, *partitioning*, and *question strategy selection*. Each step could be defined as a separate control task in the system. The rules for question strategy selection could be made explicit also, so that in each consultation cycle, one of the possible question strategies would be chosen. If *Fact-Residual Rules* were used to help specify termination criteria, then a task that applies those rules *after confirmation* of disease prototypes could be added to the cycle. The control slots of *other prototypes* could be used, as they are in CENTAUR, to represent disease-specific control knowledge. For example, this knowledge could specify the manifestations that are most critical in filling in a given prototype, or other more specific diseases to be explored once that prototype is confirmed. Explicit representation of control knowledge would allow *both systems* to explain their actions to a user, and would facilitate modifications to the data base.

REPRESENTATION IN EACH SYSTEM

STAGE	INTERNIST	PIP	CENTAUR
ENTERING INITIAL DATA	volunteered or requested manifestations	volunteered or requested findings	inferred or requested facts
TRIGGERING FRAMES	evoking strengths associated with manifestations, Constrictors	trigger findings associated with hypotheses	Triggering Rules
SCORING	multi-problem generator, scoring algorithm	scoring function represented in each hypothesis	Certainty Measure
SELECTING A FRAME	select highest ranking disease	select highest ranking hypothesis	determined by consultation strategy
FILLING IN FRAME	apply partitioning algorithm, then select questioning strategy (RULE-OUT, NARROW, PURSUE, or DISCRIMINATE)	ask about any expected finding	apply tasks in To-Fill-In slot, or if none, default procedure specified in Consultation prototype
TESTING MATCH	confirmation by numerical threshold	confirmation and exclusion lists or numerical threshold	matching criteria specified in Consultation Prototype
	↓	↓	↓
	repeat loop until all manifestations are accounted for by confirmed diseases	repeat loop until there are no unanswered questions of the active hypotheses or any of their causal relatives	apply tasks in If-Confirmed or If-Disconfirmed Slot
ACCOUNTING FOR DATA	evoking strength of manifestation for disease	findings associated with hypotheses	Plausible Values for prototypes and Fact-Residual Rules
REFINING DIAGNOSIS	(none)	(none)	Refinement Rules
SUMMARIZING AND PRINTING RESULTS	(none)	(none)	Summary Rules, Apply Tasks in Prototype Action Slots

FIGURE 6.11 Comparison of Control Structures in INTERNIST II, PIP, and CENTAUR

Chapter 7

Explanation and Knowledge Acquisition in CENTAUR

7.1 Introduction

The ability of the consultation system to give explanations of its conclusions and suitable justifications for the information it asks of users is a basic premise of work on expert systems. The ability to acquire new knowledge or modify existing knowledge is equally important for systems requiring large amounts of domain-specific knowledge that is subject to changes over time.

This chapter discusses CENTAUR's explanation system, and describes the procedures that have been implemented to aid in knowledge acquisition. EMYCIN's facility for justifying system questions is first presented and its limitations are contrasted with CENTAUR's expanded explanation capabilities and final interpretation. The role of the Agenda in understanding system processing is discussed, as well as a second top-level task that aids in understanding the knowledge base by reviewing knowledge in the stored prototypes.

7.2 Explanation in CENTAUR

In a typical consultation, the user presents a problem to the consultation system, and the system attempts to solve the problem using its own store of knowledge and additional information supplied by the user in response to system questions. At the termination of the consultation, the system's results are

presented. Understanding the consultation session includes understanding not only these final results, but also the questions being asked, the motivations for asking those questions, and the justifications for the system's intermediate conclusions. This understanding depends, in turn, on the user's ability to accept first the knowledge that is represented in the system, and second, the way in which that knowledge is used make conclusions. The essential point of this chapter is that CENTAUR's explicit representation and use of prototypical knowledge produces easily understood explanations of the consultation session.

7.2.1 Explanations of the Question

CENTAUR asks questions of the user (a) when the system needs additional information that it has failed to deduce through rules, or (b) when it needs a value for an *AskFirst parameter*.[1] When the user is asked a question, instead of supplying an answer, he can use special responses to ask for a justification for the question (the WHY and HOW options), or to see the *expected responses* to the question (the "?" option). He can also ask to see the current system control task, for instance the task, *determine the subtype of Obstructive Airways Disease*, that is causing the current sequence of questions (the CONTROL option). Part of the work *done in* CENTAUR on providing these explanations is an extension of the EMYCIN *explanation* system, discussed *briefly below*.

[1] Recall that AskFirst parameters are those whose values are asked directly of the user instead of *first attempting to infer* them with rules. (See Section 2.3).

7.2.1.1 EMYCIN Explanation System--HOW and WHY Options

The EMYCIN explanation system is used during an interactive consultation session to justify and explain the program's questions and reasoning steps.[2] The EMYCIN consultation process is essentially one of search through an AND/OR goal tree, where each node represents a goal to determine the value for some parameter needed in the consultation. If invoking a rule involves determining a new parameter value, then a new subgoal node is created. Questions are asked of the user when rules have failed to deduce the needed parameter value or when the parameter is specified as one whose *value is always obtained* from the user. Each question corresponds to a system goal, such as to determine the identity of a particular organism, and that goal corresponds, in turn, to a node in the tree.

Instead of supplying the system with the requested information, the user may *enter the word* WHY *to obtain* a justification for the question. This justification is given by reciting the rules causing the parameter value to be requested. The user may repeat the WHY question, and the system will give the next higher goal in the tree as a justification[3] WHY questions can be repeated to unravel the system's reasoning chain until the top node in the tree is reached. Each response is numbered, so that the user can ask HOW a goal was (or will be) achieved. A sample of explanations in PUFF is given in Figure 7.1. Note that all WHY and HOW inquiries

[2] The MYCIN knowledge acquisition and explanation routines were written initially by Ted Shortliffe [Shortliffe, 1976] and extended by Randy Davis [Davis, 1976]. They were adapted for EMYCIN by Carli Scott and Bill vanMelle.

[3] The WHY command has an optional argument, an integer between one and ten, that controls the size of the step taken in going up the tree. Thus the tree can *be traversed* in steps whose size is specified by the user rather than stepping from goal to goal. See [Davis, 1976] for details.

are first expanded by the system to indicate the exact interpretation of the question that the system is answering in an effort to avoid possible ambiguities in the user's inquiry. As in all of the examples, user responses are written in **boldface** capitals and are preceded by a double asterisk (**).

```
                                          (system prompt)
15) The number of pack-years of smoking:
                                          (user's request to determine
                                          motivation for question)
** WHY
                                          (system's rephrasing of question
                                          and translation of rule)
[i.e. WHY is it important to determine the
     number of pack-years smoked?]

[4.0] This will aid in determining the findings
     about the diagnosis of obstructive airways disease.
     If
        [4.1] the severity of obstructive airways disease
               is greater than or equal to mild, and
        [4.2] the number of pack-years smoked is greater
               than 0
     then
        it is definite (1.0) that Airway obstruction is
           consistent with the patient's smoking history.
           is one of the findings about the diagnosis
           of obstructive airways disease
     [RULE036]

  [back to question 15...]
                                          (The user asks HOW 4.1 above
                                          was derived.)
** HOW 4.1

[i.e. HOW is it determined whether the severity
     of obstructive airways disease is greater
     than or equal to mild?]

The following may prove useful later in the
     consultation:
          [5.1] RULE003

  [back to question 15...]
```

FIGURE 7.1 **WHY and HOW Samples from PUFF**

7.2.1.2 Problems and Limitations

EMYCIN's approach to explanation is thus to recapitulate on demand some portion of its reasoning chain, stating those rules that were used to achieve its goals. The success of this approach depends on the ability of the user to understand the chaining among rules that results in the goal structure of the consultation, and on the content and clarity of the knowledge represented in the rules themselves. Limitations and problems with the explanation facility have occurred because these two assumptions were not always met. The way in which the program reasons (the backward chaining of rules), *is not always intuitively clear* to the user. It is frequently difficult to utilize appropriate inquiries to traverse the goal tree structure and obtain a suitable explanation for a question. Problems also occur because the knowledge represented in rules is not always complete, or given at a level of detail appropriate for explanations. Further, because many rules represent context and control knowledge implicitly, explanations that provide control structure or context knowledge are not distinguished from other rule explanations.

Another limitation of the EMYCIN rule explanations is that each rule explains only the current question with no broader context given for the entire line of questioning. For example, in MYCIN when the system is invoking rules to determine which is the best drug therapy to prescribe for a patient, the rule explanations may give no indication of what infectious disease is being treated until the user has asked a sufficient number of WHYs to climb the goal tree to that higher-level goal. CENTAUR's rule explanations are given with the prototype context stated first in order to provide a perspective upon the overall line of questioning.

7.2.1.3 CENTAUR's *WHY* and *HOW* Options

CENTAUR first precedes each explanation with a statement of which prototype is currently being considered. This statement immediately sets a *context* for the more detailed explanations to follow. If a prototype has a non-zero Certainty Measure[4] then it is also given. Samples of CENTAUR's explanations are shown in Figures 7.2 and 7.3.

In CENTAUR, questions may be asked of the user during several of the consultation stages. Knowledge about stages is used in explanations to make them more understandable and to clarify the meaning of the user's *WHY* or *HOW* question. A *WHY* or *HOW* asked during one stage may be interpreted differently from the same question asked during another stage. For example, a *WHY question asked about a* prototype during initial stages of a consultation before that prototype has been confirmed is interpreted to mean *WHY are you considering this prototype as a possible match to the data?* A *WHY* question during the Refinement Stage, after the prototype has been confirmed is interpreted to mean *WHY was this prototype confirmed?*

Questions may be asked by CENTAUR as values are being sought for components of prototypes. These questions typically ask for the value of one of the components, or that of a general parameter whose value is needed for a rule, which in turn is used to infer a value for the component. When an explanation is given about a component, CENTAUR supplies the Importance Measure of that component as

[4] Prototypes originally have a Certainty Measure of zero. For some prototypes, such as those that are never triggered, this Certainty Measure will remain zero.

a part of the explanation. (See Figure 7.2.) Questions also may be asked during the final stages of the consultation when Fact-Residual, Refinement, or Summary Rules are invoked. (See Figure 7.3.)

As was stated in the previous section, in EMYCIN, multiple WHY inquiries step up the goal tree until the top node is reached. In CENTAUR, a WHY inquiry given after reaching the top node is interpreted to be an inquiry about the prototype itself, i.e., *WHY is the prototype being considered?* If the system is in its earliest hypothesis-formation stages, then information about the current hypothesis list of prototypes is first printed, followed by any parameter values that triggered the current prototype. (See Figure 7.2.) This triggering information is retrieved from the prototype *InTriggers* slot discussed in Section 5.2.6. If there are no triggering parameter values, then the information in the prototype *Origin* slot is given. A *HOW inquiry* about the prototype itself is also interpreted to mean *HOW was the prototype suggested?*, and a similar explanation is given.

```
                                        (system prompt)
 11) IVC/IVC-predicted:
                                        (user query for motivation)
 ** WHY
                                        (rephrasing of question)
[i.e. WHY is it important to determine the
    IVC/IVC-predicted ratio of PATIENT-122?]

                                        (Current prototype sets context
                                        for explanation.)

[4.0] The system is now exploring the possibility of
    Restrictive Lung Disease for PATIENT-122.   The
    current certainty for this hypothesis is .979.

                                        (A value is needed for a
                                        component of RLD.)
```

```
[4.1] The ivc/ivc-predicted ratio of a patient is one
      of the very important factors [4 on a scale of
      0 - 5] being considered for Restrictive Lung
      Disease.
       This may be useful later in the consultation.
      For example, it is relevant in determining the
      degree of restrictive lung disease as indicated
      by the IVC and is relevant in determining the
      findings about the diagnosis of restrictive
      lung disease.

[back to question 11 ...]
```

(A WHY query after reaching the top node of the goal tree is interpret.d to be a query about the prototype.)

```
   ** WHY

[i.e. WHY is the system exploring the possibility of
      Restrictive Lung Disease?]

Restrictive Lung Disease is the most strongly
      suggested hypothesis.  Other suggested
      hypotheses are:  Diffusion Defect, Obstructive
      Airways Disease.

The Restrictive Lung Disease was suggested by the
      following findings
[5.1] The tlc/tlc-predicted ratio of the patient was
          71
[5.2] The dlco/dlco-predicted of the patient was 66

[back to question 11 ...]
```

(The user asks HOW 5.1 above was determined.)

```
   ** HOW 5.1

[i.e. HOW was the TLC/TLC-predicted ratio of
      PATIENT-122 determined?]

The following were used in deducing the TLC/TLC-predicted
      ratio of PATIENT-122:
            [6.1] PULF036 indicated it is definite (1.0)
                        that the TLC(body box) of pt122 is the
                        TLC/TLC-predicted ratio of pt122

[back to question 11 ...]
```

FIGURE 7.2 WHY and HOW Queries during the initial Stages of a CENTAUR Consultation

If CENTAUR is in its refinement stage and a WHY inquiry has reached the top node in the goal tree, then the question is interpreted to mean *WHY was the prototype confirmed?* The explanation given in response to this inquiry lists the components and their values that were consistent with those expected for the prototype, and thus caused it to be confirmed. (See Figure 7.3.) As in EMYCIN, WHY responses are numbered so that the user can ask HOW that information was, or will be, determined.

(Intermediate results are presented and the consultation continues from the previous figure into its Refinement Stage.)

```
Confirmed hypotheses: Asthma, Severe Obstructive
      Airways Disease, Obstructive Airways Disease

   20) The number of pack-years of smoking:
   ** WHY

[i.e. WHY is it important to determine the number of
      pack-years smoked?]
```

(St-ge and context are explicit.)

```
[6.0] The system has entered its refinement stage and
      is executing refinement rules associated with
      Obstructive Airways Disease.

The number of pack-years smoked is used in the
      following refinement rule:
      If
         [6.1] the number of pack-years smoked is
                  greater than 0, and
         [6.2] the number of years ago that the
                  patient quit smoking is 0, and
         [6.3] the degree of smoking of 7446 is less
                  than the degree of obstructive
                  airways disease of 7446
      then
         it is definite (1.0) that the following is
            one of the conclusion statements about
            this interpretation: Smoking probably
            exacerbates the severity of the patient's
            airway obstruction.
      [PULF036]

[back to question 20 ...]
```

```
                                        (A WHY query after reaching the
                                        top node in the goal tree is
                                        interpreted to be a query about
                                        why the prototype was
                                        confirmed.)
** WHY

[i.e. WHY was Obstructive Airways Disease confirmed as being present?]

The following findings were consistent with
    Obstructive Airways Disease:
  The tlc/tlc-predicted ratio of the patient was 139
  The rv/rv-predicted ratio of the patient was 261
  The fev1/fvc ratio of the patient was 48
  The mmf/mmf-predicted ratio of the patient was 12
  The f5025 of the patient was 9
  The observed-predicted difference in rv/tlc of the
      patient was 25
  The fev1/fev1-predicted ratio of the patient was
      42

[back to question 20...]

        .
        .
        .
```

FIGURE 7.3 Sample WHY Queries during CENTAUR's Refinement Stage of Processing

7.2.1.4 The CONTROL Option

A new keyword, CONTROL, has been defined so that a user can inquire about the control task motivating the current line of reasoning. In the first example of Figure 7.4, the user is asked for the change in MMF (Maximum Mid-Expiratory Flow) after bronchodilation of the patient. The CONTROL option prints a translation of the current control task from the system agenda, in this case, determining a subtype for Obstructive Airways Disease. The response to a WHY query, on the other hand, would recite the inference rule that was being tried when the question was asked.

The control tasks are those specified in Control Slots of the prototypes. In this example, the OAD If-Confirmed Slot specifies that a subtype for OAD should be determined. In PUFF and MYCIN, this control knowledge was implicit in the inference rules and could not be given as distinct from other rule explanations. The second example in Figure 7.4 shows a control task from the Consultation Prototype being executed.

```
17) Change in MMF (after bronchodilation):
** CONTROL

The current control task is to determine the SUBTYPE of OAD.

        .
        .
        .

20) The number of pack-years of smoking:
** CONTROL

The current control task is to apply the Refinement Rules
associated with the confirmed prototypes.
```

FIGURE 7.4 Samples of the CONTROL Option

7.2.1.5 The ? Option

In EMYCIN and in CENTAUR, the user can type a "?" in response to a question to get a restatement of the question and a list of expected responses. In CENTAUR, when a question is being asked to determine the value for a prototype component, the set of expected values *in the context of that prototype* is also specified. Two examples of this option in a CENTAUR consultation are shown in Figure 7.5. In the first, the user is asked for the FEV1 (Forced Expiratory Volume in one second) of the

patient. When the "?" option is used, the system first prints the question in a slightly expanded form, followed by the set of expected responses (in this case, just the fact that a number is expected). These expected responses are associated with the general clinical parameter, FEV1, and are used to check the response given by the user to make sure that it is reasonable. Many of the PUFF parameters have numerical values with possible ranges that are very broad because they may be high in some diseases and low in others. Thus, in PUFF, it may not be possible to be more specific than to say that the expected value is a number. In CENTAUR, however, the question is asked *because one particular disease*, in this example OAD, is being considered, and within OAD, the range of expected values is more narrow. Thus a second statement of expected values for the prototype being considered can be given. The second example in Figure 7.5 shows CENTAUR's responses to the "?" query about a parameter with "Yes" and "No" as its expected answers. Because this question is asked *within the context of the prototype*, system expectations can be more clearly defined.

```
13) FEV1/FEV1-predicted:
** ?
What is the FEV1/FEV1-predicted ratio of PATIENT-7?
Expected responses are: number
Furthermore, for Obstructive Airways Disease it is expected
that the value is less than 80.

        .
        .
        .

19) Sputum purulence:
** ?
Is there sputum purulence?
Expected responses are: YES or NO
Furthermore, for Asthma it is expected that there is not
sputum purulence.
```

FIGURE 7.5 Samples of the "?" Option

7.2.1.6 Possible Extensions--General Question-Answering

EMYCIN also has a general Question-Answering facility [Scott, et al., 1977] that can be called at the end of the consultation to answer the user's questions about that particular consultation session, or about the program's store of domain-specific knowledge. Although general question-answering has not been extended for CENTAUR, it is interesting to note the additional types of questions that could be handled by a question-answering facility, building upon the representation of knowledge about prototypical cases. These include questions about the prototype control slots, such as *If Obstructive Airways Disease is confirmed, what does the system do next?* or *What does the system do in order to determine whether the patient has a Diffusion Defect?* This knowledge was not explicitly represented in MYCIN and thus was not accessible to MYCIN's general question-answerer. However, this control knowledge, as well as knowledge about the prototype components and their slots, is available in the prototype knowledge review task discussed in Section 7.4.

7.2.2 CENTAUR'S Final Interpretation

Understanding the final conclusions of a consultation system is a critical part of understanding the system's performance. CENTAUR's final interpretation of given facts is displayed at the end of the program in a more complete and comprehensible form than that given in PUFF. Recall that in PUFF, the final interpretation consists of two parts: a simple listing of different pulmonary function tests and test results for a patient, and the verbal statements that interpret the tests and give a final pulmonary

disease diagnosis. (See Section 2.5.) The form of this interpretation allows a physician to check the interpretation statements against the test results to make certain that the statements given are accurate. *Because the interpretation* statements are generated from rules, the only way for a physician to determine why a particular statement was generated is to trace back through the rule set, either by rerunning the consultation and using the WHY and HOW options, or by examining the rule base on his own. In either case, this process can be quite time-consuming, and it is sometimes not possible for the physician to have ready access to the system to utilize the on-line explanation facility.[5] PUFF essentially gives the physician only the input and the output with no justification of how that output was derived.

In CENTAUR, the simple listing of tests *and test results* has been replaced by a summary of information from the prototypes. (See Section 4.2.) Those prototypes confirmed as matching the data in a given case are printed in a format in which indentation indicates their level in the prototype network. The findings suggesting a prototype are listed, as well as those that are consistent with that prototype (the Plausible Values), inconsistent with it (the Possible Error Values or Surprise Values), and those accounted for by it (as defined in Section 6.7.1.4). Any test results that are not accounted for by some confirmed prototype are listed, including an indication of which prototypes in the knowledge base, if any, can account for these test results. Such results *provide a clue of possible errors* in the tests or of modifications needed in the knowledge base itself. Finally, a list of prototypes that were tried but disconfirmed during the consultation are printed, together with a statement of conclusions and a final diagnosis.

[5] Many of the pulmonary function interpretations are done in "batch mode" where the data are available on a tape, and the physician sees only the final interpretation.

Because the interpretation statements are generated directly from confirmed prototypes and from rules associated with confirmed prototypes, the basis for the conclusions is explicitly ascertainable from the prototype summary. More information is available about the consultation than is currently being printed. For example, for a disconfirmed prototype, the system could list those test results that were inconsistent with that prototype. Because the printing of the final interpretation is controlled by prototype control slots, the exact form and content of an interpretation can be made specific to the desires of individual physician users simply by changing the appropriate control clauses.

7.3 The Role of the Agenda in Explanation

All of CENTAUR's tasks are kept on an agenda as described *in Section* 6.5. One of the options available for running a consultation is to print tasks as they are added to the agenda and as they are selected to be executed. When a task is added to the agenda, a verbal description of the task, as well as the task's source and the reason it was placed on *the agenda, are printed.* When a task is chosen to be executed, the verbal description is printed. (See Figure 7.6.) Both printouts are available in either terse or more elaborate versions. During a consultation, a user also may request to see the next task to be executed by typing a specified control character. This feature assists the user in determining what the system is doing at any given time.

During system development, the terse form of agenda printing was extremely useful in debugging the system. Agenda traces aid in understanding the system's

control structure, and the ability to see tasks as they are being executed provides a task-oriented explanation of system behavior. The agenda printing mechanism is related to the CONTROL option discussed in Section 7.2.1.4. Both specify tasks that are being executed. The difference is that the CONTROL option is used specifically as a response to a question and explains the task causing that question to be asked. The agenda printing mechanism applies to all tasks, not just those that result in questions being asked. The following figure illustrates samples of agenda printing during a consultation.

(The system prints the next task to be executed.)

```
The next task is to confirm the current prototype.
```

(OAD is confirmed.)

```
Based on the data provided, it is confirmed that
there is Obstructive Airways Disease.
```

(When OAD is confirmed, tasks are added to the agenda from the OAD If-Confirmed Slot.)

```
Task: Determine the SUBTYPE of OAD
From Source: OAD If-Confirmed Control Slot
Added to Agenda because: Obstructive Airways Disease has
      been confirmed, so an attempt should be made to deduce
      the subtype of OAD.

Task: Determine the DEGREE of OAD
From Source: OAD If-Confirmed Control Slot
Added to Agenda because: Obstructive Airways Disease has
      been confirmed, so an attempt should be made to deduce
      the degree of OAD.
```

(The system chooses the next task.)

```
The next task is to determine the DEGREE of OAD.
    .
    .
    .
```


(Later in the consultation, a task is added to the agenda as a result of another task having been executed.)

```
Task: Order the Hypothesis List
From Source: Task adding new prototypes to the Hypothesis List
Added to Agenda because: New prototypes have been added to the
   Hypothesis List, it should be checked to see that it is
   ordered according to which prototype best fits the facts.
```

(That task is selected to be executed.)

```
The next task is to order the Hypothesis List.
    .
    .
    .
```

FIGURE 7.6 Samples of Agenda Printing During a Consultation

7.4 The Prototype Review Task

The knowledge stored in CENTAUR's prototypes gives a "typical" picture of the situation represented by each prototype. This knowledge itself is of interest to someone wanting to learn about the prototypical situation. For example, a naive user might want to learn what are the characteristic components of a typical case of Obstructive Airways Disease, or how the expert proceeds when he determines that the patient has Obstructive Airways Disease.

To allow the user to have access to this knowledge, a second top-level task to review the knowledge in the prototypes, called the **REVIEW** task, was created. The user can type the word "Review" when the system requests special options, and the

REVIEW prototype will be selected, instead of the CONSULTATION prototype, as the top-level task.

The review task requests the *name of a prototype* to be reviewed, and then presents the information specified by the IF-CONFIRMED slot of the REVIEW prototype. *The information presented* for this implementation includes the prototype components, their plausible values and importance measures, and any control knowledge associated with the prototype; for example, knowledge about what to do if the prototype is confirmed. The user can change the order in which the knowledge is presented, or choose additional sets of slots to review by modifying the control tasks in the REVIEW prototype. A portion of the review of the *OAD prototype* is *shown in Figure 7.7.*

The REVIEW prototype has no components, and two control slots. The TO-FILL-IN slot asks for the name of the prototype that is to be reviewed, and the IF-CONFIRMED slot *specifies* what information will be printed for the review. Reviews of the REVIEW prototype itself and of the CONSULTATION prototype are also shown in Figure 7.7.

```
        ***Prototype Knowledge Review***

Prototype you wish to review:
** OAD

The following are the components and expected values
    for OBSTRUCTIVE AIRWAYS DISEASE.  Importance Measures
    (from 1 to 5) are also listed.

  For the the TLC/TLC-predicted ratio of the patient it
    is expected that the value is greater than or equal to 100,
    importance measure 4.

  For the the RV/RV-predicted ratio of the patient it is
    expected that the value is greater than 140, importance
    measure 4.

  For the the FEV1/FVC ratio of the patient it is expected
    that the value is less than 80, importance measure 5.
```



```
  For the the MMF/MMF-predicted ratio of the patient it is
    expected that the value is less than 70, importance
    measure 5.

    .
    .
    .

OBSTRUCTIVE AIRWAYS DISEASE control information:

  When OBSTRUCTIVE AIRWAYS DISEASE is confirmed, the system will
     Determine the DEGREE of OAD
     Determine the SUBTYPE of OAD

Prototype you wish to review:
** REVIEW

REVIEW control information:

  To fill in REVIEW the system will
     Ask for the name of the prototype to be reviewed

  When REVIEW is confirmed, the system will
     Review the components, their expected values, and
         their importance measures
     Review knowledge in the control slots

Prototype you wish to review:
** CONSULTATION

CONSULTATION control information:

  To fill in CONSULTATION the system will
     Ask for the TRACING-LEVEL for the consultation
     Ask for the AGENDA-PRINTING for the consultation
     Ask for the STRATEGY for the consultation

  When CONSULTATION is confirmed, the system will
     Set the confirmation threshold to 0
     Set the percentage of filled-in slots necessary
         to confirm the prototype to .75
     Set the default procedure for filling in slots
         to fill in slots in decreasing order of their
         Importance Measures
     Determine the domain of the consultation
     Select the current best prototype
     Fill in the prototype
     Confirm or disprove the prototype
     Mark facts that are accounted for by the prototype
     Apply the refinement rules associated with the
         confirmed prototypes
     Apply the summary rules associated with the
         confirmed prototypes
     Execute actions associated with the confirmed
         prototypes
```

FIGURE 7.7 A Sample of the Prototype Knowledge Review Task

7.5 Explanation Conclusions

The research goal in providing CENTAUR with an explanation capability was not to create an ideal explanation facility, but rather to demonstrate possible directions for improvement given the existing EMYCIN explanation system and an altered knowledge representation and control structure. It was not necessary in CENTAUR to make extensive changes to EMYCIN's explanation system. By representing control knowledge explicitly in prototypes, however, separate control structure explanations were made possible. In addition, prototype explanations not only serve as a context within which to view detailed rule explanations, but they also aid in understanding current lines of questioning and allow sets of expected values specific to that context to be suggested as responses to questions. Additional information, such as findings consistent with a prototype, can be given. Explanations giving component *information and Importance Measures* supply some basic knowledge not represented in EMYCIN systems.

There are, of course, many more problems with both the EMYCIN and CENTAUR explanation systems that are beyond the scope of this research. These include multiple interpretations of the WHY or HOW inquiries,[6] lack of a user model to guide the level of detail given in an explanation, and restriction to an explanation that depends directly on the knowledge representation and control structure of the system, instead of providing a separate explanation facility.

---

[6] For example, does the user mean *WHY is the system asking me for this piece of information instead of some other piece of information?* or *WHY is the system asking me this information in the current context instead of in some other context?*

7.6 Knowledge Acquisition in CENTAUR

7.6.1 The Initial Set of Prototypes

Representing knowledge in prototypes has not been found to be difficult. This is partly because many medical texts present diseases in terms of typical cases so that the knowledge is already close to prototype form. Unlike rule interactions that necessitate understanding the way rules will be invoked, prototype interactions are minimal, and are specified explicitly in slots associated with each prototype so that the expert can supply this information separately. In particular, the static relationship of one prototype with others in the prototype network is specified in pointer slots: MORE SPECIFIC, MORE GENERAL, and ALTERNATE.

Further, acquiring prototypes *first* may actually facilitate the acquisition of rules. It has been our experience that experts working with us have had difficulty formulating a set of specialized rules to represent a new disease area without the added structure of these typical disease patterns. In both the cystitis and pulmonary function domains, experts first defined a set of typical disease patterns and then were more easily able to write specialized rules to handle more rare situations, or to recommend therapy or further tests.

7.6.2 Acquiring and Modifying Prototypes

To facilitate adding new prototypes to the knowledge base and modifying existing prototypes, CENTAUR includes a function called PROTOTYPEMAKER, which interacts with the person who maintains the knowledge base (the "knowledge

engineer"), to acquire the modifications and update the appropriate knowledge representation structures. This function is intended to be used by someone who is familiar with the CENTAUR system, not by the domain expert. Its purpose is to simplify the acquisition and maintenance of the domain knowledge. The prototypes are represented internally as LISP record structures. (See Appendix D for an example.) PROTOTYPEMAKER serves as a helpful interface, allowing the knowledge engineer to represent domain knowledge without having to store it directly in its final LISP form. This section presents an example of a session with PROTOTYPEMAKER. As before, user input is in **BOLDFACE CAPITALS** preceded by a double asterisk (**), and comments about the interaction are in *italics*.

(The interaction begins as the knowledge engineer calls PROTOTYPEMAKER from within LISP. The "←" is the LISP prompt.)

```
←PROTOTYPEMAKER]
```

(The user has three options: to add a new prototype, modify an existing prototype, or delete a prototype from the knowledge base.)

```
Please select from among  ADD, DELETE, MODIFY
Prototype Option:
** MODIFY
Prototype Name:
** SUPER-NORMAL
Slot Name:
** ?
```

(PROTOTYPEMAKER will list possible slot names when "?" is typed. Possible component names, and component slots are also listed when "?" is typed to the corresponding prompts.)

```
Slot names are: NAME HYPOTHESIS EXPLANATION AUTHOR DATE
SOURCE MOREGENERAL MORESPECIFIC ALTERNATE INTRIGGERS ORIGIN
CM COMPONENTS TO-FILL-IN ACTION REFINEMENT-RULES SUMMARY-RULES
IF-CONFIRMED IF-DISCONFIRMED FACT-RESIDUAL-RULES
```

(The user decides to add a new component, called LOCALE.

PROTOTYPEMAKER now prompts the user for all of the component slot information. In each case, the prototype name, component name, and slot name are printed first to orient the user, and to clarify what information is being requested.)

```
Slot:
** COMPONENTS

Please select from among  ADD, MODIFY, DELETE
Component Option:
** ADD
Component Name:
** LOCALE
```

(It is not necessary to give every slot a value. OK means that the current value, in this case NIL, is satisfactory. Any non-NIL current value is printed first to inform the user before changes are made.)

```
SUPER-NORMAL
LOCALE
DEFAULT VALUE:
** OK

SUPER-NORMAL
LOCALE
PLAUSIBLE VALUES:
** ((POLLUTED-ENVIRONMENT))

SUPER-NORMAL
LOCALE
POSSIBLE ERROR VALUES:
** (((EQ $VALUE NORMAL-ENVIRONMENT) (OUTTRIGGER NORMAL 500)))
```

(The current value is printed here for the user. All components are initialized with an Importance Measure of 0.)

```
SUPER-NORMAL
LOCALE
IMPORTANCE MEASURE:
Current Value: 0
** 3
```

(The user may list any existing inference rules when the component is created. However, rules that update an existing component are automatically added to this list as is described in Section 7.6.4.)

```
SUPER-NORMAL
LOCALE
INFERENCE RULES:
** (RULE180 RULE181)
```

(The user now decides to print the SUPER-NORMAL prototype using a second function, PRINTPROTOTYPES, to see the changes he has made. Only the LOCALE component is shown in this figure. PRINTPROTOTYPES is the same function used to print the OAD prototype in Appendix D and the Higher-level prototypes in Section 6.4.)

```
←←PRINTPROTOTYPES((SUPER-NORMAL))

NAME:  SUPER-NORMAL
    .
    .
    .

COMPONENT:  LOCALE

PLAUSIBLE VALUES:
If: Polluted-environment
Then: no action indicated

POSSIBLE ERROR VALUES:
If: The value is normal-environment
Then: Suggest NORMAL with a certainty measure of 500

IMPORTANCE MEASURE:  3

INFERENCE RULES:  RULE180 RULE181

*****************************************************************
```

(Now returning to PROTOTYPEMAKER, the user decides to add a new prototype, called CYSTITIS. This interaction shows the CYSTITIS MOREGENERAL slot being given a value.)

```
    .
    .
    .
Prototype Option:
** ADD
Prototype Name:
** CYSTITIS
New prototype CYSTITIS created.
```

(The user specifies that he wants to fill in the MOREGENERAL slot of the prototype.)

```
Slot Name:
** MOREGENERAL
```

(Again, the prototype name and slot name are printed first to orient the user.)


```
CYSTITIS
MOREGENERAL
** (DOMAIN INFECTIOUS-DISEASE)
```

(The user has noticed an error in the syntax of the MOREGENERAL slot he has just specified. PROTOTYPEMAKER allows the user to edit slot values by interfacing with the INTERLISP editor.)

```
Slot Name:
** MOREGENERAL
CYSTITIS
MOREGENERAL
Current Value: (DOMAIN INFECTIOUS-DISEASE)
```

(The user gives the special EDIT command.)

```
** EDIT
edit
(DOMAIN INFECTIOUS-DISEASE)
```

(The single asterisk is the editor's prompt. The BI is an edit command to surround the first and second elements of the current expression with parentheses.)

```
*(BI 1 2)
```

(P is the print command.)

```
*P
((DOMAIN INFECTIOUS-DISEASE))
```

(OK is the command to leave the editor.)

```
*OK
```

7.6.3 Modifying the Structure of Prototypes

CENTAUR also provides the user with auxiliary functions to update the existing set of prototypes in the event that a modification is made to the basic prototype structure, for instance to add a new type of slot or to delete an existing type. If a structural change in the prototypes is desired, the system designer must first edit the prototype *record structure* (specified in Appendix C) using the INTERLISP record editor. The system designer then calls a CENTAUR function, **UPDATE-PROTOTYPES,**

which modifies each of the existing prototypes to make them consistent with the new record structure. There is also a second function, **UPDATE-COMPONENTS**, which similarly updates the existing components when the component record structure has been modified.

7.6.4 Adding or Modifying Rules in CENTAUR

An auxiliary function provided by EMYCIN, called **GETRULES**, is used to add or *modify rules in the knowledge base.* EMYCIN allows the system designer to write a second function, called **GETRULEUSERFN**, which is called by GETRULES whenever a modification is made to the knowledge base of rules. GETRULEUSERFN thus is used to perform additional bookkeeping or updating of the knowledge base. The GETRULEUSERFN defined in CENTAUR updates the rule slots in prototypes whenever a new rule is added or deleted, or when the function or prototype associated with an existing rule is changed. Changing the function of the rule, for example from an Inference Rule to a Refinement Rule, requires deleting the rule number from the appropriate Inference Rule slot and adding it to the Refinement Rule slot in the same prototype. Changing the prototype associated with a rule requires deleting the rule from one prototype slot and adding it to the corresponding slot in the second prototype.

CENTAUR's GETRULEUSERFN *also prompts the user for the* PROTOTYPE property[7] for any new rules, and keeps a record of all changes that are made on an auxiliary

[7] Recall that the PROTOTYPE property of a rule is a pointer to the prototype *with which the rule is associated, and is used for purposes of explanation when the* rule is examined independently of the prototype.

file. This section presents a sample session with GETRULES. Comments are in *italics* and the user's responses are in **BOLDFACE**.

(GETRULES is called from LISP. In this session, the user decides to create a new Fact-Residual Rule.)

```
←GETRULES]

Rule #, NEW or subject for new rule: FACT-RESIDUAL RULE
Antecedent rule? No
```

(The user is given a new rule number, RULE160, and is prompted for the PREMISE, ACTION, and PROTOTYPE properties of the new rule.)

```
RULE160
PREMISE: ($AND (GREATEQ* (VAL1 CNTXT TLC) 120))
RULE160
ACTION: (DO-ALL (MARKFACT TLC SUPER-NORMAL)
        (CONCLUDETEXT CNTXT FINDINGS-CONC (TEXT
        $SN1 "The high total lung capacity is consistent
        with super-normal pulmonary function.") TALLY 1000))
RULE160
PROTOTYPE: NORMAL
```

(The user chooses to see the English translation of the new rule.)

```
Translate, No further change, or prop name: TRANSLATE

RULE160
-------

  If:  The tlc/tlc-predicted ratio of the patient is
       greater than or equal to 120
  Then:  1) Mark the TLC as being accounted for by
            SUPER-NORMAL, and
         2) It is definite (1.0) that the following
            is one of the conclusion statements about
            this interpretation: The high total lung
            capacity is consistent with super-normal
            pulmonary function.

  PROTOTYPE:  NORMAL

Translate, No further change, or prop name: NO
```

Chapter 8

Summary and Conclusions

This final chapter reviews the major themes of the research presented in this dissertation, and discusses questions posed by the choice of representation structures. It notes CENTAUR's role as an "AI applied to Medicine" (AIM) system, and presents the results of testing CENTAUR's performance on a set of actual patient cases, in comparison both with practicing physicians and with PUFF. CENTAUR is analyzed as a second-generation AI system, and some observations are offered from experiences in acquiring the same knowledge in two different forms from a single expert. This chapter also discusses CENTAUR's usefulness as a tool for experimenting with variations in control of the consultation.

8.1 Review of Major Themes

In preceding chapters, a set of desirable characteristics for knowledge representation structures used in large knowledge-based systems has been described. The effectiveness of representing prototypical knowledge for performing consultations has been emphasized in particular. The major themes of this research can be grouped roughly into four categories, as follows: knowledge representation, control, knowledge acquisition, and explanation.

Knowledge Representation:

First, it is important that the chosen knowledge representation structures be *expressive enough to represent a variety of types of knowledge* about the domain, such as both knowledge of prototypical cases and more specific inferential knowledge. This may necessitate using more than one type of knowledge structure, as was done in CENTAUR with prototypes and production rules. Using more than one knowledge structure to encode the different types of domain knowledge has the added advantage of *separating the different types of knowledge*. It may be possible to represent all of the domain knowledge in one type of knowledge structure, but if this approach compromises some of the other criteria for knowledge representations expressed in this thesis, then any savings resulting from choosing a single representation will be costly in other respects. This is not to say, however, that there should be a proliferation of knowledge structures; using the same structure when possible to encode different *levels* of knowledge allows the system to use the same set of routines for all levels, as discussed in [Davis, 1976]. This principal is followed in CENTAUR where the prototype structure is used both for representation of the high-level consultation and knowledge review problems and also for domain-level pulmonary diseases. Explanation and knowledge acquisition routines for domain-level prototypes thus can be used for high-level prototypes as well.

A second theme is that *information associated with domain*

knowledge, such as the context in which the knowledge is applied, and the purpose or function of the knowledge in the consultation, must be represented explicitly with that knowledge. The explicitness of this associated information is critical in all parts of the consultation--from the context provided for consultation questions themselves to the explanations or justifications of system conclusions. Other information, such as the inherent reliability of the data, should be represented explicitly and separately from the inferential knowledge used to form system conclusions.

Third, it is necessary to represent *expected patterns of data for each context* in the consultation in order to accurately detect inconsistent or erroneous data and to be able to offer assistance in context to a user answering consultation questions. It is not enough to give only general lists of expected values, as is done in PUFF, because this does not enable a user to determine what answer is expected in a particular problem-solving context, and it does not give the system the ability to detect data that are inconsistent in terms of the current problem-solving context. There *are additional benefits to classifying data according to* prototypical patterns, such as being able to store and retrieve cases according to the patterns they match, in order to test changes in the system or to serve as examples of system expertise in specific areas.

Finally, the hierarchical arrangement of knowledge from general to specific categories in CENTAUR allows more specific prototypes, rules,

and components to *inherit knowledge* from their more general counterparts. *Inheritance* of knowledge is beneficial for two reasons. First, knowledge common to several *specific entities* can be represented a single time in a more general entity. Second, although the most specific knowledge is applied where possible, general knowledge can be applied in situations where more specific knowledge has failed.

■ Control:

An important theme for control is that *control knowledge must be represented explicitly*, and separately from inferential knowledge. One of the problems that occurred in PUFF's rule representation was the confusion caused by control knowledge being represented implicitly in inference rules. This made it difficult to determine interactions between rules, and made it necessary for the user to understand the backward-chaining of rules in order to modify the rule base. CENTAUR's representation of control knowledge in slots associated with each prototype allows context-specific control to be clearly specified. This explicit representation facilitates modifications to the control knowledge.

Another control theme is that *questions asked during the consultation should be sensitive both to the initial set of data, and to the context that is being explored.* The control of the consultation should, therefore, allow the initial data to suggest the most likely contexts to explore, and the contexts, in turn, should guide further search for

information. Because PUFF explored diseases in a fixed order as specified in rules, questions were not always sensitive to the initial pulmonary function test results, and without broad contexts to guide more detailed processing, irrelevant questions frequently were asked. A system that represents contexts for questions explicitly also can inform the user of the context being explored, and can print intermediate conclusions to indicate progress being made during the consultation.

In addition it is important to compare the way the program reasons with the way human experts reason. *There is an intuition reflected by* many researchers in the medical domain (e.g., [Kassirer and Gorry, 1978], [Szolovits and Pauker, 1978], [Pople, 1977]) that a program whose reasoning more closely parallels physicians' reasoning will be more easily understood and accepted by physician users. Although it is not the contention of this thesis that the control structure for CENTAUR represents the way physicians reason, it does appear that CENTAUR's overall control structure, one which suggests likely diseases to explore on the basis of the initial data and guides further processing on the basis of prototypical knowledge, does in fact more closely resemble physicians' reasoning than does the backward chaining of rules in EMYCIN systems. The ability to understand the conclusions of a consultation system, which depends in part on understanding the reasoning steps used to derive those conclusions, is essential to the user's acceptance of the system and of its conclusions.

- **Knowledge Acquisition:**

What is required of a method of knowledge representation intended to simplify the process of knowledge acquisition? The knowledge must be organized in such a manner that it is easy to locate and modify. CENTAUR's explicit representation and organization of its knowledge base into groups of knowledge dealing with prototypical situations facilitates knowledge acquisition. The prototypes represent blocks of basic knowledge, and include clearly defined "hooks" for any additional rules necessary to elaborate upon this basic knowledge. The purpose of the knowledge attached to these hooks is explicit, making the effect of such modifications readily predictable.

- **Explanation:**

The ability to explain system performance has become a critical factor in the acceptance by users of large knowledge-based consultation systems. When system explanations are generated from the actual representations used for the performance program, then the quality of those explanations depends upon the completeness and explicitness with which the performance knowledge is represented. Any explanations that justify conclusions by replaying the control processes that directed those conclusions also must depend on the understandability of the control structure. Although better explanations might be obtained from a system whose explanatory knowledge is represented separately from its

performance knowledge. Such systems would have the added burden of having to modify their explanatory knowledge whenever changes were made to their performance knowledge.

CENTAUR has demonstrated that when a system has explicit and complete representations of its knowledge and an understandable control structure, reasonable explanations can be generated directly from the performance knowledge.

8.2 Rules or Frames or Both?

This thesis has presented two systems that perform the task of pulmonary function interpretation. In PUFF, domain-specific knowledge is represented as a single set of inference rules. The control structure for the production rule representation simply tests conditions in the premise clauses of rules and, when those conditions are true, forms the conclusions specified in the action clauses. This research has explored the problems that arose from attempting to represent all of the domain knowledge in rules, and has illustrated in CENTAUR a system using both rules and frames to represent knowledge explicitly. CENTAUR's control structure is more sophisticated, enabling the system to do more than simply test conditions and form conclusions. In summarizing experiences gained from this research, the following questions dealing with representation and control choices need to be addressed. *What are the desirable features of the expert system that were difficult to achieve using the production rule representation alone? Could PUFF be rewritten to avoid its problems without the addition of frames to its data base? How about*

creating an all-frame system to perform the same task? Would other representations, for instance semantic nets, do as well?

First, some of the features that are desirable in an expert system, which were not present in the rule-based systems, include the following:

1) Representation of knowledge as patterns of data typically encountered in the domain,

2) Classification of actual data patterns in terms of prototypical data patterns,

3) Representation of (possibly overlapping) ranges of plausible data values for each prototypical data pattern, and

4) Use of data cues to suggest probable directions for further search.

5) Separation of domain expertise to be applied at different stages during processing.

Frames are more suitable structures than rules for representing *standard data patterns with ranges of plausible data values* specified for each frame. Further, these ranges of values can be overlapping from one frame to the next. For example, a TLC value between 80 and 120 is considered plausible for a patient with Normal pulmonary function, and a TLC value of more than 100 is plausible for a patient with Obstructive Airways Disease. A TLC value between 100 and 120 is thus consistent with both Normal pulmonary function and Obstructive Airways Disease. Similarly, in determining the degree of a disease, the boundary between mild and moderate disease or between moderate and moderately-severe disease can be blurred by

representing overlapping ranges of expected data values. In rule representations, where rules make clear-cut decisions about each degree indicator in sequence, an overlap in expected values is not possible. For example, in order to determine the degree of Restrictive Lung Disease, one rule specified a degree in terms of ranges of values for the TLC measurement, so that a TLC of less than 50 was Severe, between[1] 50 and 60 was Moderately-severe, between 60 and 70 was Moderate, and between 70 and 100 was Mild. In practice, however, it would be impossible to draw so sharp a line between the degrees. A TLC of 59 might indeed indicate Moderately-severe Restrictive Lung Disease, but it would also be a reasonable value for Moderate Restrictive Lung Disease, especially if the other data values indicated a "moderate" degree.

CENTAUR's control structure has a number of capabilities not present in the production rule systems. It was designed to utilize *data clues* not only to suggest likely prototypes to explore initially, but also to suggest alternative hypotheses while processing information during the consultation. For example, possible error values associated with a component slot may suggest other more likely prototypes when information fails to match.

Separate sets of expertise are applied during different stages of processing in CENTAUR by grouping the production rules according to their function and applying them at different points in the consultation. To do this in the rule-based systems is difficult at best, because there is no division of knowledge according to those points during the consultation when it should be applied. In fact, the rule-based systems

[1] Between is defined here to include the lower limit, but not the upper limit.

exhibit only a single inference stage in which all inference rules are applicable. In order to approximate the stages of a CENTAUR consultation, the lists of inference rules associated with parameters (and used to determine values for them) have to be ordered. For example, a sort of "Refinement Stage" of processing can be produced by placing rules mentioning the same parameter both in their premise and their conclusion (termed *Self-referencing Rules*) last in the rule lists. The form of these rules is generally as follows: *if one believes that X is true and Y also is true, then one can be more certain that X is true.* These rules are a form of refinement knowledge, *and comment on an interim* conclusion (*that X is true*), in order to help form a final conclusion. The order of rules in the rule lists forces all of these Self-referencing rules to be executed *after* the other rules. Unfortunately, this control knowledge is implicit and must be told to system designers in order to ensure that such refinement knowledge will be properly executed.

It is important that the chosen data structures be expressive enough to represent a variety of types of knowledge. If a system is restricted to a single representation structure, however, much of the information in the resulting knowledge base may be implicit, making it more difficult for other researchers or users to understand or modify the *knowledge base*. *One problem* of the constrained rule representation used in PUFF was that it was difficult to represent prototypical data patterns in rules. All of the rules shared the same format of premise and action clauses, and all were treated in the same way by the rule-handling routines in the system. This did simplify processing, but this also caused many of the representation problems discussed in Chapter 3.

It is also critical to represent knowledge *explicitly according to its function* in the consultation. Prototypes afford a means of representing explicit contexts, control and consultation stages, as well as explicit error handling and other types of knowledge. One basic problem in PUFF is that a *single set* of rules was used to represent all of the knowledge. A rule-based system that indexes rules in a variety of ways, for example, according to their function in the consultation (control rules, inference rules, summary rules, etc.), and according to the context in which they applied (OAD rules, RLD rules, etc.), would make some of this knowledge explicit. If this were attempted in the EMYCIN rule-based systems, the present rule-handling routines would have to be made significantly more complicated in order to deal with such altered rule sets.

An all-frame representation for this problem would be possible, with the production rule knowledge mapped into frames. The premise clauses of a production rule could be represented in one frame linked to a second frame representing the rule's conclusion. Such an all-frame system is in fact being implemented experimentally for the pulmonary function interpretation problem.[2] It also is possible to represent rule knowledge in semantic nets as has been done in the PROSPECTOR system [Duda et al., 1979].

8.3 CENTAUR as an AIM System

CENTAUR belongs to a *family of medical AI* systems and has made several contributions in this "AI applied to medicine" (AIM) domain. One of the goals of AIM

[2] This system is being developed at Stanford by Dave Smith and Jan Clayton.

systems is to represent enough of physicians' expertise to be able to deal effectively with medical problems. CENTAUR has demonstrated that it is feasible to use *multiple sets of knowledge* to capture various aspects of physicians' expertise in order to solve problems in a medical domain. PUFF, MYCIN, PIP, and INTERNIST II, among others, represent physicians' expertise in inferring new information in the domain, but CENTAUR represents other expertise as well, for example, expertise dealing with data discrepancies (Fact-Residual Rules), and diagnosis *refinement* (Refinement Rules). More importantly, the explicit representation of all of this expertise makes it easier to modify and add new expertise to the knowledge base.

The representation of expected data patterns as prototypes allows CENTAUR to detect inconsistent or erroneous data frequently encountered in medical domains, with the result that the system will not pursue unproductive lines of reasoning, and the user can be alerted to such data. A user also can specify actions to be taken by the system for "Possible Error Values" encountered during a consultation.

CENTAUR has demonstrated the benefits of using all known data to further constrain the search for additional information. This includes using initial data to suggest the most likely diseases (Triggering Rules), and using knowledge about which are the most important components of each disease to guide further questioning. It is important for all AIM systems to focus search efforts and to perform consultations in as short a time as possible.

CENTAUR has shown that representation of control *knowledge for disease* contexts is an important part of physicians' expertise about the domain, and that it is useful as a means to guide the program's reasoning, and to explain the physicians'

AD-A091 177   STANFORD UNIV CA DEPT OF COMPUTER SCIENCE   F/G 9/2

PROTOTYPES AND PRODUCTION RULES: A KNOWLEDGE REPRESENTATION FOR--ETC(U)

AUG 80 J S AIKINS   MDA903-77-C-0322

UNCLASSIFIED STAN-CS-80-814   NL

2 OF 2

AD-

A091177

END

DATE

FILMED

12 80

DTIC

methodologies for dealing with various diseases. Further, by representing control knowledge as part of the domain-specific knowledge, different control strategies can be represented for different physicians. Thus, the control structure, like the inference knowledge, can be tailored to individual physician users.

Parts of CENTAUR's design have been incorporated into a new AIM system, the ONCOCIN system now under development at Stanford.[3] ONCOCIN is a consultation system designed to advise *physicians* at the Stanford Hospital Oncology Day Care Center on the management of cancer patients undergoing experimental treatment protocols. Most of ONCOCIN's domain knowledge is represented using production rules, and two "prototype-like" representation structures, **Contexts** and **Control Blocks**. The Contexts in ONCOCIN represent static knowledge about basic entities in the domain, such as the diseases and protocols. *The Control Blocks are discrete* ordered sets of steps used to accomplish specific tasks, such as calculating the correct dosage of a drug. These structures afford in ONCOCIN the same advantages in the explicit representation of context and control knowledge as have been discussed for prototypes in CENTAUR. Further, in ONCOCIN as in CENTAUR, production rules are specifically associated with the Contexts in which they are applied.

8.4 CENTAUR as a Second Generation AI System

As discussed in Section 1.6, CENTAUR is one of a small number of "second generation" AI systems--systems that have been designed to perform the same task in the same domain as a first generation system, but which vary some fundamental

[3] ONCOCIN is being developed by Carli Scott, Miriam Bischoff, and Ted Shortliffe.

design feature such as the knowledge representation or control structure. Second generation systems are beneficial because they can demonstrate clear advances in the field of AI through *comparisons* with their *first* generation counterparts. Their contribution is not in accomplishing a new task, but establishing a new *method* for accomplishing tasks. This can be done as in CENTAUR in the manner of a laboratory experiment, by keeping certain parts of both systems constant, and varying others to determine the effect on the overall system.

In CENTAUR and PUFF, the initial data and final interpretation were constant, *but by changing the knowledge representation* and control structure, a much improved task performance resulted. Furthermore, many of the benefits achieved in CENTAUR's performance were even more *evident for explanation and knowledge* acquisition. This thesis has dealt primarily with the improved performance and explanation capabilities of CENTAUR over PUFF. *Although it is difficult to precisely measure* the *qualitative* improvement in knowledge acquisition, the experience in developing these two knowledge bases is illustrative.

Representing much of the domain knowledge in a new form in CENTAUR offered an opportunity to compare the ease with which a single expert was able to express *his expertise* in two different forms. We were fortunate to have an expert who was willing to give his expertise both in rules for PUFF, and again later in prototypes for CENTAUR. *We first* introduced him to the production rule formalism when we began gathering the rule knowledge for PUFF. He learned to give us his expertise in rules rather quickly, and we were able to transform them almost *directly into inference* rules. The primary difficulties in acquiring the initial set of knowledge for PUFF were

in setting up "control" rules to guide the execution of other inference rules. The expert was pleased with the modularity of the rule set which allowed him to add or modify rules rather easily. As the rule set grew, however, it became more difficult for him to reconstruct the reasons for certain rule clauses (especially those originally inserted for control purposes), and a lack of organization of the rules around disease types made it difficult to view the set of expertise for each disease collectively.

When he was introduced to the concept of prototypes for CENTAUR, the expert immediately located an existing list of standard clinical findings for each pulmonary disease. (Such lists are readily available in medical texts.) These lists were the basis for the initial set of prototypes. It was also easier for the expert to review the knowledge and to make modifications, because the expertise for each disease was associated explicitly with a prototype *representing that disease.*

When we explained CENTAUR's control process of using the initial data to suggest likely hypotheses and exploring these hypotheses further on the basis of such data, he remarked "That's how I think." We make no attempt here to substantiate his claim, but can add that the hypothesize and test paradigm as manifested by CENTAUR seemed easier for him to understand than the backward chaining used in PUFF.

The additional sets of expertise such as the Fact-Residual rules which dealt with data discrepancies, and the Refinement rules which recommended additional tests or made the final decision about which diseases accounted for findings in the final interpretation, were created in response to expertise that the expert applied to real cases that we simulated during development of the knowledge base.

8.5 Development and Validation of the Knowledge Base

CENTAUR's original knowledge base was tested on ten cases chosen from a file of cases in the pulmonary function laboratory at Pacific Medical Center in San Francisco. Those ten cases formed a representative sample of the various pulmonary diseases, their degrees and subtypes. Modifications were made to the knowledge base and the ten cases were tried again. This iteration continued until the expert was satisfied that the system's interpretations agreed with his own. At this point the system was frozen and a new set of 100 cases was selected and interpreted by the system. All of the 100 cases were also interpreted separately by PUFF and by two pulmonary physiologists (the expert working with us and a physician from a different medical center).

The results of the comparison of interpretations by each diagnostician are presented in the table in Figure 8.1. The table compares "close" agreement in diagnoses of the severity of the disease, where "close" is defined as differing by at most one degree of severity. Thus, for example, two diagnoses of severity, mild (degree=1) and moderate (degree=2), are close, while mild and severe (degree=3) are not. Further, a diagnosis of *normal* is not considered to be close to a diagnosis of a mild degree of any disease.

The table shows that that the overall rate of agreement between the two physiologists on the diagnoses of disease was 84%, and the agreement between PUFF and CENTAUR was 87%. The agreement between the two systems and the physician who served as the expert to develop them (MD-2 in the table) was 85% for PUFF and 91% for CENTAUR. Finally, the agreement between the two systems

and the physician who had no part in their development (MD-1) was 74% for PUFF and 84% for CENTAUR. PUFF has no representation of knowledge about neuromuscular disease (NMD), and so no comparisons were made with PUFF on NMD diagnoses (as indicated by the asterisks in the table). Figure 8.2 shows the distribution of diagnoses by each diagnostician.

To analyze the significance of the results, a chi-squared test with one degree of freedom (as defined in [Brown, 1977]) was performed on the percent of agreement of CENTAUR and PUFF with each physician. The test showed that there is a statistically significant difference in the percent of agreement between the two systems and MD-1 ($P < .001$), and between the two systems and MD-2 ($P < .01$). That is, there is less than a .001 chance that there is no statistical difference in the two measurements in the first case, and less than .01 in the second.

Thus it appears that the physician who served as the expert for developing the two systems was able to achieve a better match of his own knowledge to the prototype representation than he was to the rule representation. The outside expert also agreed more with the prototype system than the rule system. In fact, the outside expert agreed as often with CENTAUR as he did with the other physician.

	PERCENT AGREEMENT					
DIAGNOSIS	MD-1 MD-2	MD-1 PUFF	MD-2 PUFF	MD-1 CENTAUR	MD-2 CENTAUR	PUFF CENTAUR
NORMAL	98	89	95	98	96	99
OAD	84	82	88	82	98	85
RLD	86	79	88	85	89	88
DD	69	48	71	71	82	75
NMD	93	*	*	94	99	*
TOTAL (S.D.)	84 (1.6)	74 (2.2)	85 (1.8)	84 (1.6)	91 (1.3)	87 (1.7)

FIGURE 8.1 Summary of Percent Agreement in 100 Cases

	DIAGNOSTICIAN			
DIAGNOSIS	MD-1	MD-2	PUFF	CENTAUR
NORMAL	19	25	28	21
OAD	62	58	58	59
RLD	27	35	35	28
DD	15	37	57	33
NMD	1	7	*	2

FIGURE 8.2 Number of Diagnoses by Each Diagnostician for 100 Cases

8.6 Consultation Comparisons

Measurements of the number of questions asked during a consultation and the order in which the information was obtained were made in order to permit comparisons between CENTAUR and PUFF. A representative sampling of twenty cases has shown that, because CENTAUR first explores the most likely diseases, as indicated by the data, the questions are asked in a slightly different order. After some initial information is obtained by both systems, PUFF always begins asking questions about OAD, followed by RLD, Diffusion Defect, and the possibility of a Normal diagnosis, as indicated by the PUFF "goal" rule shown in Figure 2.4. The order of CENTAUR questions is similar to the order of PUFF questions only in cases where OAD is the most likely disease. Further, in many cases, CENTAUR does not ask some of the questions asked by PUFF because it has found no evidence to indicate that some of the diseases even should be considered.

The numbers of questions asked in each PUFF and CENTAUR consultation is another indication of each program's ability to focus on the most relevant information. In the twenty cases tested, CENTAUR asked, on the average, four questions fewer per consultation than did PUFF, when only the pulmonary disease concepts common to both systems were considered. When considering actual numbers of questions asked, in all but one of the twenty cases sampled, CENTAUR asked more questions than PUFF, because more pulmonary disease concepts are represented in CENTAUR than in PUFF. For example, CENTAUR represents concepts (such as the F25) dealing with air flow through the lungs, as well as those dealing with lung volumes which are represented in PUFF. The air flow knowledge in CENTAUR serves as a second set of

evidence used to diagnose disease. Other concepts, such as Sputum Purulence, which helps to distinguish Asthma from Bronchitis, were added by our expert in developing the prototypes. The knowledge available to both systems can be equalized by answering CENTAUR's additional questions with the response "UNKNOWN". This was done in comparing the diagnoses on the 100 cases just discussed. The additional knowledge in that comparison also was not available to the two pulmonary experts.

CENTAUR also asks some questions in its refinement stage that deal with concepts not known to PUFF. If we discount these "refinement" questions, and those questions answered UNKNOWN, then the modified system, CENTAUR*, does indeed ask fewer questions per consultation. The results of this comparison are presented in Figure 8.3, with the last column showing the savings in numbers of questions asked between PUFF and the modified CENTAUR* system.

CASE NUMBER	DIAGNOSES	PUFF QUESTIONS	CENTAUR QUESTIONS	CENTAUR* QUESTIONS	P – C* DIFFERENCE
1	D	16	15	12	4
2	O	13	22	11	2
3	N	13	15	12	1
4	O,R,D	17	25	11	6
5	R,D	15	19	10	5
6	D	13	14	11	2
7	R,D	13	16	12	1
8	D	13	14	11	2
9	O,R,D	16	23	11	5
10	O,D	17	21	11	6
11	O,R,D	17	24	12	5
12	O,R,D	15	23	11	4
13	O,R,D	18	21	12	6
14	N	13	14	11	2
15	D	13	15	12	1
16	O	17	23	10	7
17	D	15	16	11	4
18	D	13	14	10	3
19	O	16	21	11	5
20	O,R,D	17	24	10	7
MEAN (S.D.)		15.0 (1.8)	18.9 (4.1)	11.1 (0.7)	3.9 (2.0)

(Diagnoses: O=OAD, R=RLD, N=Normal, D=Diffusion Defect)

FIGURE 8.3 Number of Questions Asked Per Consultation

8.7 CENTAUR as a Tool for Experimenting with Consultations

One possible use for the CENTAUR system is as a tool to permit experimentation with various control schemes for consultations. CENTAUR's prototypes provide a

convenient representation of the consultation process itself, with control of the consultation being represented in the control slots. This research has already shown that by representing the consultation stages as tasks in the control slots, variations on the basic control scheme, such as running the consultation with and without a refinement stage, can be performed.

One experiment performed with CENTAUR was to run a set of twenty cases on each of the three consultation strategies, *confirmation*, *elimination*, and the *fixed-order* strategy. The results of this experiment are presented in the table in Figure 8.4. Each entry in the table shows the number of questions asked during the consultation (in parentheses), and the number of prototypes tried and confirmed or tried and disconfirmed (as a ratio of the number confirmed over the number disconfirmed). The last two columns indicate which strategy or strategies, if any, resulted in the fewest prototypes being explored, and the fewest consultation questions being asked (C=Confirmation, E=Elimination, and F=Fixed-order). The confirmation strategy (attempting to confirm the most *likely prototype*) resulted in the fewest questions and exploration of the fewest prototypes, whenever any strategy was a clear winner. The fixed-order strategy (exploring suggested prototypes in a pre-set order) was the second best strategy, and sometimes tied with the confirmation strategy in minimizing numbers of questions and prototypes. The elimination strategy (attempting to eliminate the least likely prototype) was the worst, and never won over either of the other two strategies. The fixed-order used in this experiment placed the disease prototypes in order of their *a priori* probability for patients at Pacific Medical Center. The cases in which the fixed-order strategy tied with the confirmation strategy were those in which the actual diagnosis of disease matched this *a priori* list.

CASE NUMBER	FIXED-ORDER STRATEGY	ELIMINATION STRATEGY	CONFIRMATION STRATEGY	FEWEST PROTOS	FEWEST QUESTS
1	5/3 (25)	5/5 (25)	5/0 (25)	C	
2	1/0 (15)	1/0 (15)	1/0 (15)		
3	1/1 (19)	1/1 (19)	1/1 (19)		
4	1/3 (19)	1/4 (19)	1/3 (19)	C F	
5	5/4 (24)	5/8 (24)	5/0 (24)	C	
6	2/1 (22)	2/2 (22)	2/0 (20)	C	C
7	4/0 (23)	4/7 (24)	4/0 (23)	C F	C F
8	1/1 (21)	1/2 (21)	1/1 (21)	C F	
9	3/1 (24)	3/4 (24)	3/1 (24)	C F	
10	3/1 (23)	3/6 (23)	3/1 (23)	C F	
11	3/3 (21)	3/6 (23)	3/0 (21)	C	C F
12	1/0 (14)	1/0 (14)	1/0 (14)		
13	2/2 (21)	2/2 (21)	2/0 (15)	C	C
14	3/0 (23)	3/2 (23)	3/0 (23)	C F	
15	2/1 (16)	2/1 (16)	2/1 (16)		
16	2/0 (21)	2/4 (22)	2/0 (21)	C F	C F
17	4/3 (24)	4/6 (25)	4/0 (24)	C	C F
18	3/3 (22)	3/8 (24)	3/0 (22)	C	C F
19	4/1 (19)	4/4 (19)	4/0 (15)	C	C
20	1/0 (15)	1/0 (15)	1/0 (15)		

Notation: Prototypes confirmed / Prototypes disconfirmed (Questions Asked)

C=Confirmation, E=Elimination, F=Fixed-order

FIGURE 8.4 Comparison of Consultation Strategies

The consultation strategy experiment was performed chiefly to demonstrate the flexibility of CENTAUR's control representation, but experiments like these are also useful in answering research questions such as *which is the most efficient strategy?*, and *which strategy is preferred by most users?* Variations on other points of control, such as testing different prototype confirmation thresholds, could be the subject of future experiments. Because these points of control are defined explicitly in prototype control slots, such experiments are easily performed.

Experiments with the knowledge base include adding new high-level task prototypes, like the CONSULTATION and REVIEW prototypes, to perform additional tasks in the pulmonary disease domain. For example, a *Retrospective Analysis* task that considers an initial set of data and a final interpretation and attempts to form a reasoning path between them, might well be considered.

Another experiment would be to add additional sets of domain prototypes to perform consultations in other domains. CENTAUR's knowledge representation and control structure is not dependent upon the pulmonary function domain, and in fact, initial sets of prototypes for both the meningitis and cystitis domains were formulated early in the development of the system to test the system's generality, that is, for extending it to other domains.

Experiments such as these are a critical dimension in Artificial Intelligence, where building expert systems is still more often an art than a science. The CENTAUR system has demonstrated its usefulness as a tool for experimenting with choices of knowledge representation structures and control schemes, and its potential for building new consultation systems in domains where prototypical knowledge can be used to guide problem solving. *It is important that future expert systems be as flexible in their design as is CENTAUR, to enable them to adapt through experimentation to new tasks and new domains.*

Appendix A

Glossary of Medical Terms

Sources for the information in this glossary are [Cherniack, 1977] and [West, 1974].

Asthma
A subtype of Obstructive Airways Disease marked by paroxysms of difficult breathing, wheezing, and cough.

Body Plethysmography (or Body Box Method)
A physical method for measuring the total lung capacity (TLC) of the patient (and other lung volume measurements) in which the patient sits in a large airtight box, or body plethysmograph, and breathes into a mouthpiece. This measurement technique is simple and reliable, and is particularly useful because the flow resistance of the airways can be determined at the same time. (See also *Gas Dilution Method*).

Bronchitis
A subtype of Obstructive Airways Disease characterized by inflammation of the bronchial tubes.

Bronchodilation
Dilation of the bronchi, useful in detecting Asthma and in determining whether the respiratory disease has a reversible component.

Diffusion Defect
Impaired capacity of the lung to transfer gas from air spaces to the red cells in the pulmonary capillaries.

DLCO Diffusing Capacity for Carbon Monoxide
The ability of the lungs to transfer carbon monoxide from the alveolar air into pulmonary capillary blood.

Dyspnea
Shortness of breath.

Emphysema
A subtype of Obstructive Airways Disease characterized by overdistention of air spaces in the lung.

F25
The rate of expiratory air flow at 25 per cent of vital capacity (VC).

F50
The rate of expiratory air flow at 50 per cent of vital capacity (VC).

F5025
The slope of the flow-volume curve between 50 and 25 per cent of vital capacity (VC).

$$F5025 = (F50 - F25) / FVC$$

FEV1 Forced Expiratory Volume in one second
The volume of air expelled in one second during a forced expiration, starting at full inspiration, i.e., at total lung capacity (TLC).

Flow-Volume Curve
The plotted relationship between the rate of air flow and the volume change during a forced vital capacity maneuver.

FVC Forced Vital Capacity
The volume of air expired during a rapid forced expiration starting at full inspiration, i.e., at total lung capacity (TLC) and ending at residual volume (RV).

Gas Dilution Method (or DLCO Method)
Another method for measuring the patient's total lung capacity. The patient inspires and expires from a spirometer in either single breath or multiple breath procedures and in either a closed (rebreathing) or an open (non-rebreathing) system. This is used as alternative to the *Body Plethysmograph Method*. The dilution techniques also provide information about the distribution of ventilation.

MMF Maximum Mid-expiratory Flow rate
The mean rate of expiratory air flow between 25 and 75 per cent of the forced expiratory vital capacity.

Neuromuscular
Pertaining to nerves and muscles (as in *Neuromuscular Disease* or *NMD* which is characterized by a low TLC and early flow rates being equal to or less than later flow rates).

OAD (Obstructive Airways Disease)
A category of lung diseases characterized by obstruction to the flow of air in and particularly out of the lung.

Pack-Years
A measure of the smoking history of the patient calculated as the number of packs of cigarettes smoked each day multiplied by the number of years the patient has been smoking.

Purulent
Consisting of or containing pus (as in *sputum purulence*).

RLD (Restrictive Lung Disease)
A pattern of abnormal lung function defined by a decrease in lung volume.

Glossary of Medical Terms

RV Residual Volume
The volume of air remaining in the lungs after a maximal expiration.

Thus RV = TLC - VC

Spirometer
An instrument for measuring the air taken into and exhaled from the lungs.

Supernormal Pulmonary Function
A pattern characterized by high lung volumes in an otherwise normal patient.

TLC Total Lung Capacity
The sum of all the compartments of the lung, or the volume of air in the lungs at maximum inspiration.

VC Vital Capacity
The maximum volume of air that can be expelled after a maximum inspiration, i.e., from total lung capacity (TLC).

Appendix B

General Control Tasks

This appendix lists the general control tasks used during the consultation process with a description of each one. The tasks are implemented as functions in Interlisp. The task descriptions are text strings stored with each task and used to explain system actions, for example when tasks that are being executed from the agenda are described.

Task Name	Task Description
ACCOUNTFOR-FACTS	Mark the facts which are accounted for by this prototype.
CHECK-ACTION-SLOT	Add the tasks in the Action Slots of the *confirmed* prototypes to the agenda.
CHECK-CONFIRMED-LIST	Check to see if the *current* prototype is already confirmed.
CHECK-FACT-MATCH	Check to see if the known facts match the current prototype.
CHECK-FACT-POOL	Check to see if there are any remaining facts that have not been accounted for by a confirmed prototype.
CHECK-HYPOTHESIS-LIST	Check the hypothesis list for possible prototypes to explore.
CHECK-TO-FILL-IN-SLOT	Add the tasks in the To-Fill-In Slot of the current prototype to the agenda.
CHECK-TRIGGERED-LIST	Check to see if there are any triggered prototypes.
CONFIRM-PROTOTYPE	Confirm the current prototype, and add tasks in the If-Confirmed Slot to the agenda.
FILL-IN-NEW-INFO	Fill in components of all relevant prototypes with the new facts.

General Control Tasks

FILL-IN-NEW-PTS	Fill in components of new prototypes with the already known facts.
FIND-OUT-COMPONENTS	Determine if there are components to be filled in for the current prototype.
ORDER-HYPOTHESIS-LIST	Order the Hypothesis List according to the prototype Certainty Measures.
SELECT-CURRENT-PT	Select a prototype from the Hypothesis List according to the current consultation strategy.

Appendix C

Templates for Data Structures

This appendix presents the templates for the three frame-like data structures used in the CENTAUR system: Prototypes, Components, and Facts. Each template is a *record declaration* in Interlisp [Teitelman, 1978], and each instance of the templates is an Interlisp *record structure*. Names are associated with the various parts or *fields* of the records. Auxiliary functions have been written for CENTAUR to allow easy access to the data in these fields. Default values for a field can be set in the record declaration itself, as is done for the numerical fields in each record declaration below, and for the AUTHOR and DATE fields of the prototype record declaration.

Prototype Template:

```
(RECORD PROTOTYPE
        (NAME
         HYPOTHESIS
         EXPLANATION
         AUTHOR
         DATE
         SOURCE
         MOREGENERAL
         MORESPECIFIC
         ALTERNATE
         INTRIGGERS
         ORIGIN
         CERTAINTY-MEASURE
         MATCH-MEASURE
         TO-FILL-IN
         IF-CONFIRMED
         IF-DISCONFIRMED
         ACTION
         REFINEMENT-RULES
         SUMMARY-RULES
         FACT-RESIDUAL-RULES
```

```
        COMPONENTS)
    AUTHOR ←'AIKINS
    DATE ←(DATE)
    CERTAINTY-MEASURE ← 0
    MATCH-MEASURE ← 0)
```

Component Template:

```
(RECORD COMPONENT
        (CNAME
         DEFAULT-VALUE
         PLAUSIBLE-VALUES
         POSSIBLE-ERROR-VALUES
         IMPORTANCE-MEASURE
         INFERENCE-RULES
         ACTUAL-VALUE)
    IMPORTANCE-MEASURE ← 0)
```

Fact Template:

```
(RECORD FACT
        (FNAME
         FACT-VALUE
         CERTAINTY-FACTOR
         WHERE-FROM
         HOW-CLASSIFIED
         ACCOUNTED-FOR)
    CERTAINTY-FACTOR ← 1000)
```


Appendix D

OAD Prototype and Components

This appendix gives the actual Obstructive Airways Disease prototype in both its LISP and English versions. The English version is generated automatically from the LISP version using templates stored with the functions in the same manner as has been described for the rules. (See Section 2.2.2.) In the LISP version which is presented first, the slot names are specified in *italics* in the left-hand margin for reference. Some of the slot values are NIL, either because they have not been specified for this prototype (such as the values for the *Alternate* slot and the *To-Fill-In* slot), or because they are filled in with values during the consultation (such as the *InTriggers* and *Origin* slots). The templates for the prototypes and components were presented in Appendix C.

Name
    (OAD
Hypothesis
    "there is Obstructive Airways Disease"
Explanation
    "Obstructive Airways Disease"
Author
    AIKINS
Date
    "27-OCT-78 17:13:29"
Source
    FALLAT
MoreGeneral
    ((DOMAIN PULMONARY-DISEASE))
MoreSpecific
    ((SUBTYPE ASTHMA) (SUBTYPE BRONCHITIS) (SUBTYPE EMPHYSEMA)
    (DEGREE MILD-OAD) (DEGREE MODERATE-OAD)
    (DEGREE MODERATELY-SEVERE-OAD) (DEGREE SEVERE-OAD))
Alternate
    NIL
InTriggers
    NIL

Origin
NIL
Certainty Measure
0
Match Measure
0
To-Fill-In
NIL
If-Confirmed
((DETERMINE DEGREE OAD) (DETERMINE SUBTYPE OAD))
If-Disconfirmed
NIL
Action
((ONCEKNOWN CNTXT FINDINGS-OAD T)
(PRINTCONCLUSIONS CNTXT FINDINGS-OAD T)
(CONCLUDETEXT CNTXT FINDINGS-SUMMARY (TEXT $DEGOAD (VAL1 CNTXT DEG-OAD) "Obstructive Airways Disease") 1000))
Refinement Rules
(RULE036 RULE038 RULE039 RULE040 RULE041 RULE042 RULE043 RULE045 RULE047 RULE049 RULE085 RULE086 RULE088 RULE142 RULE143 RULE144 RULE145)
Summary Rules
(RULE053 RULE054 RULE055 RULE083 RULE090 RULE165 RULE166 RULE168)
Fact-Residual Rules
(RULE157 RULE158 RULE159)
Components: Name, Default Value, Plausible Values, Possible Error Values, Importance Measure, Inference Rules, Actual Value

```
((TLC NIL (((GREATEQ* $VALUE 100))) NIL 4 NIL NIL)
(RV NIL (((GREATERP* $VALUE 140))) NIL 4 NIL NIL)
(FEV1/FVC NIL (((LESSP* $VALUE 80))) NIL 5 NIL NIL)
(MMF NIL (((LESSP* $VALUE 70))) NIL 5 NIL NIL)
(F5025 NIL (((LESSP* $VALUE 50))) NIL 3 NIL NIL)
(F25 NIL (((LESSP* $VALUE 60))) NIL 5 NIL NIL)
(D-RV/TLC NIL (((GREATEQ* $VALUE 5))) NIL 5 NIL NIL)
(FEV1 NIL (((LESSP* $VALUE 80))) NIL 5 NIL NIL)
(D-FEV1/FVC NIL (((GREATEQ* $VALUE -5))) NIL 5 NIL NIL)
(RDX NIL ((($OR (SAME CNTXT RDX OAD)
   (SAME CNTXT RDX ASTHMA) (SAME CNTXT RDX EMPHYSEMA)
   (SAME CNTXT RDX BRONCHITIS)))) NIL 0 NIL NIL)
(COUGH NIL ((ANYVALUE)) NIL 1 NIL NIL)
(SPUTUM NIL ((ANYVALUE)) NIL 1 NIL NIL)
(REVERSIBILITY NIL ((ANYVALUE)) NIL 0 (RULE019 RULE020
   RULE022 RULE025) NIL)))
```


PRINTPROTOTYPES is the name of the function used to generate and print the English version of the LISP prototypes. NIL values are not printed by this function.

←(PRINTPROTOTYPES '(OAD))

NAME: OAD
HYPOTHESIS: There is obstructive airways disease.
EXPLANATION: Obstructive Airways Disease
AUTHOR: AIKINS
DATE: 27-OCT-78 17:13:29
SOURCE: FALLAT
MOREGENERAL: (DOMAIN PULMONARY-DISEASE)
MORESPECIFIC: (SUBTYPE ASTHMA) (SUBTYPE BRONCHITIS) (SUBTYPE EMPHYSEMA) (DEGREE MILD-OAD) (DEGREE MODERATE-OAD) (DEGREE MODERATELY-SEVERE-OAD) (DEGREE SEVERE-OAD)
CERTAINTY MEASURE: 0
MATCH MEASURE: 0

IF-CONFIRMED:
Determine the DEGREE of OAD
Determine the SUBTYPE of OAD

ACTION:
An attempt has been made to deduce the findings about the diagnosis of obstructive airways disease
Display the findings about the diagnosis of obstructive airways disease
There is evidence that the following is one of the summary statements about this interpretation:
<deg-oad> Obstructive Airways Disease

REFINEMENT RULES: RULE036 RULE038 RULE039 RULE040 RULE041 RULE042 RULE043 RULE045 RULE047 RULE049 RULE085 RULE086 RULE088 RULE142 RULE143 RULE144 RULE145

SUMMARY RULES: RULE053 RULE054 RULE055 RULE083 RULE090 RULE165 RULE166 RULE168

FACT RESIDUAL RULES: RULE157 RULE158 RULE159

COMPONENTS

COMPONENT NAME: TLC
PLAUSIBLE VALUES:
If: The value is greater than or equal to 100
Then: no action indicated
IMPORTANCE: 4

COMPONENT NAME: RV

PLAUSIBLE VALUES:
  If: The value is greater than 140
  Then: no action indicated
IMPORTANCE: 4

COMPONENT NAME: FEV1/FVC
PLAUSIBLE VALUES:
  If: The value is less than 80
  Then: no action indicated
IMPORTANCE: 5

COMPONENT NAME: MMF
PLAUSIBLE VALUES:
  If: The value is less than 70
  Then: no action indicated
IMPORTANCE: 5

COMPONENT NAME: F5025
PLAUSIBLE VALUES:
  If: The value is less than 50
  Then: no action indicated
IMPORTANCE: 3

COMPONENT NAME: F25
PLAUSIBLE VALUES:
  If: The value is less than 60
  Then: no action indicated
IMPORTANCE: 5

COMPONENT NAME: D-RV/TLC
PLAUSIBLE VALUES:
  If: The value is greater than or equal to 5
  Then: no action indicated
IMPORTANCE: 5

COMPONENT NAME: FEV1
PLAUSIBLE VALUES:
  If: The value is less than 80
  Then: no action indicated
IMPORTANCE: 5

COMPONENT NAME: D-FEV1/FVC
PLAUSIBLE VALUES:
  If: The value is greater than or equal to -5
  Then: no action indicated
IMPORTANCE: 5

COMPONENT NAME: RDX
PLAUSIBLE VALUES:
  If: 1) Oad is one of the referral diagnosis of the patient,

    2) Asthma is one of the referral diagnosis of the patient,
    3) Emphysema is one of the referral diagnosis of the patient, or
    4) Bronchitis is one of the referral diagnosis of the patient
  Then: no action indicated
IMPORTANCE: 0

COMPONENT NAME: COUGH
PLAUSIBLE VALUES:
  ANYVALUE
IMPORTANCE: 1

COMPONENT NAME: SPUTUM
PLAUSIBLE VALUES:
  ANYVALUE
IMPORTANCE: 1

COMPONENT NAME: REVERSIBILITY
PLAUSIBLE VALUES:
  ANYVALUE
IMPORTANCE: 0
INFERENCE RULES: RULE019 RULE020 RULE022 RULE026

References

The following abbreviations are used in the Reference section.

BBN Bolt, Beranek, & Newman. Cambridge, MA.

IJCAI4 *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, USSR, September 1975 (available from The Artificial Intelligence Laboratory, Publications Department, 545 Technology Square, Cambridge, MA 02139).

IJCAI5 *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Mass., August 1977 (available from Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 16213).

IJCAI6 *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, August 1979 (available from Computer Science Department, Stanford University, Stanford, CA 94305).

SRI SRI International. Menlo Park, CA.

[Bobrow, et al., 1977] Bobrow, D. G., Kaplan, R., Kay, M., Norman, D., Thompson, H., and Winograd, T. GUS, A Frame-driven Dialog System. *Artificial Intelligence*, 1977, *8*(2), 155-173.

[Bobrow and Winograd, 1977] Bobrow, D. G., and Winograd, T. An Overview of KRL, A Knowledge Representation Language. *Cognitive Science*, 1977, *1*(1), 3-46.

[Brachman, 1978] Brachman, R. J. *A Structural Paradigm for Representing Knowledge*. BBN Report 3605, May 1978.

[Brown, 1977] Brown, B. W. *Statistics: A Biomedical Introduction*. New York: John Wiley and Sons, 1977.

[Buchanan and Feigenbaum, 1978] Buchanan, B. G., and Feigenbaum, E. A. DENDRAL and Meta-DENDRAL: Their Applications Dimension. *Artificial Intelligence*, 1978, *11*(1,2), 5-24

[Cherniack, 1977] Cherniack, R. M. *Pulmonary Function Testing*. Philadelphia: W. B. Saunders Company, 1977.

[Clancey, 1979] Clancey, W. *Transfer of Rule-Based Expertise through a Tutorial Dialogue*. STAN-CS-79-769, Stanford University, September 1979.

[Davis, 1976] Davis R. *Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases*. STAN-CS-76-552, Stanford University, July 1976.

[Davis and Buchanan, 1977] Davis, R., and Buchanan, B. G. Meta-level Knowledge: Overview and Applications. *IJCAI5*, 1977, pp. 920-927.

[Davis, Buchanan, and Shortliffe, 1977] Davis, R., Buchanan, B. G., and Shortliffe, E. H. Production Rules as a Representation for a Knowledge-based Consultation Program. *Artificial Intelligence*, 1977, *8*(1), 15-45.

[Davis and King, 1977] Davis R., and King, J. An Overview of Production Systems. In E. W. Elcock and D. Michie (Eds.), *Machine Intelligence 8*. New York: Wiley & Sons, 1977. Pp. 300-332.

[Duda, et. al., 1979] Duda, R. O., Hart, P. E., Konolige, K., and Reboh, R. *A Computer-based Consultant for Mineral Exploration*. SRI Project 6415, Final Report, September 1979.

[Duda, et al., 1978] Duda, R. O., Hart, P. E., Nilsson, N. J., and Sutherland, G. L. Semantic Network Representations in Rule-Based Inference Systems. In D. A. Waterman and F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems*. New York: Academic Press, 1978. Pp. 203-221.

[Elstein, Shulman, and Sprafka, 1978] Elstein, A. S., Shulman, L. S., and Sprafka, S. A. *Medical Problem Solving--An Analysis of Clinical Reasoning*. Harvard University Press, Cambridge, Mass., 1978.

[Ernst and Newell, 1969] Ernst, G. and Newell, A. *GPS: A Case Study in Generality and Problem Solving*. New York: Academic Press, 1969.

[Goldstein and Roberts, 1977] Goldstein I. P., and Roberts, R. B. NUDGE, A Knowledge-based Scheduling Program. *IJCAI5*, 1977, pp. 257-263.

[Hewitt, 1972] Hewitt, C. *Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot*. MIT AI TR 258, MIT, April 1972.

[Kassirer and Gorry, 1978] Kassirer J. P. and Gorry, G. A. Clinical Problem Solving: A Behavioral Analysis. *Annals of Internal Medicine 89*: 245-255, 1978.

[Kunz, et al., 1978] Kunz, J., Fallat, R., McClung, D., Osborn, J., Votteri, B., Nii, H., Aikins, J., Fagan, L., and Feigenbaum, E. *A Physiological Rule Based System for Interpreting Pulmonary Function Test Results.* HPP-78-19 (Working Paper), Heuristic Programming Project, Dept. of Computer Science, Stanford University, December 1978.

[Lenat, 1976] Lenat, D. B. *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search.* STAN-CS-76-570 (AIM-286), Stanford University, July 1976.

[Lindsay, et al., 1980] Lindsay, R., Buchanan, B. G., Feigenbaum, E. A., and Lederberg, J. *DENDRAL.* New York: McGraw-Hill, 1980.

[Lusted, 1968] Lusted, L. B. *Introduction to Medical Decision Making.* Springfield, Illinois: Charles C. Thomas, 1968.

[McCracken, 1979] McCracken, D. Representation and Efficiency in a Production System for Speech Understanding. *IJCAI6*, 1979, pp. 556-561.

[Minsky, 1975] Minsky, M. A Framework for Representing Knowledge. In P. Winston (Ed.), *The Psychology of Computer Vision.* New York: McGraw-Hill, 1975. Pp. 211-277.

[Newell, 1973] Newell, A. Artificial Intelligence and the Concept of Mind. In R. Schank and K. Colby (Eds.), *Computer Models of Thought and Language.* San Francisco: W. H. Freeman and Company,1973. Pp. 1-60.

[Pauker et al., 1976] Pauker, S. G., Gorry, G., Kassirer, J., and Schwartz, W. Towards the Simulation of Clinical Cognition: Taking a Present Illness by Computer. *The American Journal of Medicine*, June 1976, *60*, 981-996.

[Pauker and Szolovits, 1977] Pauker, S. G., and Szolovits, P. Analyzing and Simulating Taking the History of the Present Illness: Context Formation. In Schneider/Sagvall Hein (Eds.), *Computational Linguistics in Medicine.* Amsterdam: North-Holland Publishing Company, 1977. Pp. 109-118.

[Pople, 1975] Pople, H. E. DIALOG: A Model of Diagnostic Logic for Internal Medicine. *IJCAI4*, 1975, pp. 848-855.

[Pople, 1977] Pople, H. E. The Formation of Composite Hypotheses in Diagnostic Problem Solving--An Exercise in Synthetic Reasoning. *IJCAI5*, 1977, pp. 1030-1037.

[Scott, et al., 1977] Scott, A. C., Clancey, W. J., Davis, R., and Shortliffe, E. H. Explanation Capabilities of Production-based Consultation Systems. *American Journal of Computational Linguistics*, 1977, *Microfiche 62.*

[Shortliffe, 1976] Shortliffe, E. H. *MYCIN: A Rule-based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection.* Ph. D. dissertation in Medical Information Sciences, Stanford University, 1974. (Also, *Computer-Based Medical Consultations: MYCIN.* New York: American-Elsevier, 1976.

[Shortliffe and Buchanan, 1975] Shortliffe, E. H., and Buchanan, B. G. A Model of Inexact Reasoning in Medicine. *Mathematical Biosciences*, 1975, *23*, 351-379.

[Szolovits and Pauker, 1976] Szolovits, P. and Pauker, S. G. Research on a Medical Consultation System for Taking the Present Illness. *Proceedings of Third Illinois Conference on Medical Information Systems*, 1976.

[Szolovits and Pauker, 1978] Szolovits, P. and Pauker, S. G. Categorical and Probabilistic Reasoning in Medical Diagnosis. *Artificial Intelligence*, 1978, *11*(1), 115-144.

[Swartout, 1977] Swartout, W. A Digitalis Therapy Advisor with Explanations. *IJCAI5*, 1977, pp. 819-825.

[Teitelman, 1978] Teitelman, W. *INTERLISP Reference Manual.* Xerox Palo Alto Research Center, Palo Alto, Ca., October 1978.

[vanMelle, 1980] Vanmelle, W. *EMYCIN: A Domain-independent Production-rule System for Consultation Programs.* (Forthcoming Ph. D. Thesis), Heuristic Programming Project, Dept. of Computer Science, Stanford University, 1980.

[Waterman, 1970] Waterman, D. A. Generalization Learning Techniques for Automating the Learning of Heuristics. *Artificial Intelligence*, 1970, *1*, 121-170.

[West, 1974] West, J. B. *Respiratory Physiology--The Essentials.* Baltimore: The Williams and Wilkins Company, 1974.